Departamento de Telemática

TRABAJO FIN DE GRADO
Grado Ing. Tecnologías de Telecomunicación

Implementación del protocolo SCVP en dispositivos móviles Android

Autora: Silvia Pajares Avendaño
Directora: Florina Almenares Mendoza
Tutora: Patricia Arias Cabarcos

Leganés, Septiembre de 2014

Título: IMPLEMENTACIÓN DEL PROTOCOLO SCVP EN DISPOSITIVOS MÓVILES ANDROID.
Autor: Silvia Pajares Avendaño
Director: Florina Almenares Mendoza

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Me gustaría sobre estas líneas mostrar mi agradecimiento a todas aquellas personas que, de un modo u otro, han contribuido con su cariño y apoyo a que pueda concluir una de las etapas más importantes de mi vida.

En primer lugar, y de una manera muy especial, quería dedicarles este trabajo a mis padres por su apoyo incondicional durante todo este tiempo. Gracias por todos vuestros consejos, por vuestra confianza y comprensión, y por recordarme la importancia del esfuerzo y el afán de superación en cada momento. Gracias por todos los valores que me habéis inculcado. Sois y seréis siempre un ejemplo a seguir.

Gracias a ti también Javi, por haber sido una fuente de optimismo y entusiasmo.

En general gracias a toda mi familia por creer en mí y en particular a ti Lucía, por animarme siempre a dar lo mejor.

Por supuesto a Alberto, por haberme acompañado a lo largo de este camino. Gracias por haberme escuchado y apoyado durante de todo este tiempo. Gracias por confiar en mí de manera incondicional.

A todos mis amigos y compañeros de clase, por vuestra ayuda y los buenos momentos vividos. En definitiva, esta etapa no hubiera sido lo mismo sin vosotros.

Y por último quería agradecer a mi tutora Patricia su disponibilidad y ayuda a lo largo del proyecto.

La gratitud da sentido a nuestro pasado, trae
paz al presente y crea una visión para el mañana

Resumen

En la actualidad, la firma electrónica y los mecanismos de autenticación basados en infraestructuras de clave pública se encuentran en pleno auge para incrementar la seguridad en el uso de servicios *online*. Esa necesidad de seguridad ha provocado que el uso de certificados digitales se haya extendido de manera notable en multitud de gestiones y transacciones. Sin embargo, puesto que éste es un mercado emergente, el uso de dichos certificados en sistemas móviles se encuentra todavía en pleno desarrollo.

Este proyecto tiene como objetivo proporcionar la implementación del protocolo SCVP (*Server-based Certificate Validation Protocol*) descrito en la RFC 5055. Dicho protocolo permite al cliente delegar el descubrimiento de la ruta de certificación y/o la validación de certificados digitales *X.509* a un servidor, por lo que resulta especialmente útil para dispositivos con recursos limitados.

Aunque el primer borrador del protocolo cuenta con más de once años, actualmente no existen implementaciones no comerciales de SCVP que proporcionen la funcionalidad de construcción y/o validación de rutas de certificación.

La aplicación propuesta, para la implementación del protocolo, está diseñada para funcionar en dispositivos móviles que cuenten con Android, el sistema operativo móvil más utilizado en el mundo.

El sistema implementado interactúa con el cliente mediante una aplicación instalada en el dispositivo móvil cuya conexión al servidor le permite delegar, las ya mencionadas tareas de validación o construcción de la ruta de un certificado.

En definitiva, el proyecto presenta la implementación del protocolo SCVP para lo cual se ha desarrollado, por un lado un cliente basado en sistema operativo Android con su correspondiente interfaz de usuario y por otro lado, un servidor.

SCVP, DPD, DPV, Android, X.509

Abstract

Nowadays, the digital signature and authentication mechanisms based on the public key infrastructures are growing to increase the security of online services. This need for security has increased the requirement for digital certificates in many processes and transactions. However, as it is a new market, the use of such certificates in mobile systems is still to be developed fully.

The aim of this project is to implement the SCVP (*Server-based Certificate Validation Protocol*) protocol described in the RFC 5055. That protocol allows a user to delegate certification path construction and certification path validation to a server, which makes it especially useful for devices with limited resources.

Even if the first draft of the protocol is eleven years old, currently there are not non-commercial SCVP implementations that provide the path construction and path validation functionalities.

The proposed application, for the protocol implementation, is designed to operate in mobiles devices with Android, the most widely used mobile operating system.

The implemented system interacts with the user through an application installed on the mobile device. Its connection to the server allows the user to delegate the task of validating or creating the path certificate.

Ultimately, the project is about an implementation of the SCVP protocol which has been developed based on an Android client with its corresponding interface, and a server.

SCVP, DPD, DPV, Android, X.509

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Contexto socio-económico y Motivación

En la actualidad, el *smartphone* se ha convertido en un elemento indispensable en la vida tanto personal como profesional de las personas. Las funcionalidades que ofrecen cada uno de ellos y la independencia proporcionan se perfilan como las cualidades mejor valoradas por los usuarios.

Somos los usuarios, los que cada vez más, buscamos que nuestros *smartphones* sean capaces de realizar las mismas acciones que realizaría un PC (*Personal Computer*). De ellos, se necesita que además de tener un tamaño reducido, sean cada vez más rápidos en sus funciones. Es decir, buscamos que nuestro teléfono móvil nos permita realizar cualquier transacción o gestión *online* sin necesidad de desplazarnos.

Todo ello unido, a una necesidad incipiente de preservar la confidencialidad e integridad de los datos en el tratamiento de la información, hacen que las gestiones *online* requieran continuamente mayores garantías de seguridad.

Por lo tanto, este entorno socio-económico actual propicia que la tecnología de la información, cada vez de una manera más clara, se encuentre sujeta a normas, leyes o reglamentos de seguridad haciendo que dicha seguridad se haya desarrollado en los últimos 30 años hasta convertirse en una de las áreas de investigación más importantes de la informática.

Las transacciones bancarias, facturaciones, gestiones administrativas con el Ministerio de Hacienda o cualquier otro trámite que anteriormente se realizaba burocráticamente y presencialmente para la correcta identificación de la persona, han cambiado su modo de operación hacia la era *Internet*. De esta forma, el comercio electrónico y otro tipo de transacciones *online* se encuentran, ahora, en pleno desarrollo.

La manera de suplir, con garantías, la identificación de la persona para aquellas acciones que incurran en un riesgo o pongan en peligro la seguridad de los datos del cliente, es mediante el uso de certificados digitales.

Estos certificados, son en definitiva, un documento electrónico otorgado por una autoridad reconocida de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa. Dichos certificados son la base para operaciones de criptografía pública ya que permiten establecer un enlace entre una clave pública y su propietario.

Las principales ventajas de los certificados es que cuentan con la capacidad de firmar o cifrar documentos electrónicamente con garantías de seguridad. Sin embargo, hoy en día, el número de aplicaciones móviles desarrolladas que hagan uso de dichos certificados es escaso.

De acuerdo a todos los datos expuestos anteriormente y teniendo en cuenta que Android está situado como uno de los sistemas operativos móviles más utilizados del mundo, resulta interesante que no se haya implementado ninguna aplicación que permita a un usuario móvil delegar la validación de un certificado digital a un servidor, para su posterior uso desde dicho dispositivo.

Puesto que el proceso de validación de un certificado consume recursos de memoria, procesamiento y batería, resulta útil la delegación de la operación de validación en dispositivos móviles debido a que estos cuentan con recursos limitados.

Por lo tanto, la motivación de este proyecto es la implementación del protocolo SCVP (*Server-based Certificate Validation Protocol*) de modo que se permita a un usuario conectarse a un servidor para el descubrimiento de la ruta de certificación o la validación del certificado.

1.2 Objetivos

El objetivo final de este proyecto es aportar una solución real en materia de seguridad, que adaptándose a las necesidades tanto profesionales como personales de las personas, permita la correcta validación de los certificados digitales. Para ello, el reto fundamental pasa por introducir, en dispositivos móviles con sistema operativo Android, la funcionalidad para la validación de dichos certificados fundamentada en el protocolo SCVP.

Clarificada la finalidad fundamental del proyecto, se infieren una serie de objetivos a alcanzar en base a dicha meta. Estos objetivos pasan por el análisis del funcionamiento

del protocolo SCVP, el estudio de la librerías y APIs (*Application Programming Interface*) necesarias para el desarrollo de las actividades criptográficas del sistema o la implementación, tanto en el cliente como en el servidor, de la lógica asociada a la construcción y/o validación de rutas.

Por lo tanto, el sistema implementado permitirá al usuario la validación de certificados *X.509* desde su dispositivo móvil. Para lo cual, el sistema contará con un módulo servidor, que basado en el protocolo SCVP será el encargado de realizar las tareas de validación, así como con un módulo cliente basado en una interfaz gráfica que realizará la interacción entre el usuario y dicho servidor. En este último módulo, será donde se le permita al usuario configurar los parámetros de validación así como adjuntar el certificado en cuestión.

En definitiva, la disponibilidad de utilización, es decir la validez de un certificado digital, siempre respaldado en los valores criptográficos básicos de autenticidad, confidencialidad, integridad y no repudio, son los principales objetivos sobre los que se fundamenta el desarrollo del proyecto.

1.3 Contenido de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo:

- En el **Capítulo 2** , "Estado del arte", se introducen las distintas tecnologías utilizadas en el desarrollo del proyecto. Primero se realiza un estudio sobre el proceso de validación de certificados digitales y el formato estándar de certificados digitales *X.509*. En segundo lugar se presenta el protocolo *SCVP* y su correspondiente funcionamiento. En tercer lugar se procede al estudio de sistemas distribuidos basados en la arquitectura cliente-servidor. Y por último se realiza una explicación detallada del sistema operativo Android y la API de seguridad *Bouncy Castle* utilizada en el proyecto.

- En el **Capítulo 3**, "Descripción general del sistema", se realiza la descripción del sistema desglosada en la arquitectura, funcionalidad y diseño de la aplicación. Asimismo se incluyen la especificación de requisitos del sistema y los casos de uso.

- En el **Capítulo 4**, "Implementación del sistema", se recogen las características de la interfaz del usuario diseñada así como la lógica interna del sistema.

- En el **Capítulo 5**, "Pruebas", se documentan las pruebas realizadas en cada módulo de la aplicación reportando resultados y otros datos de interés.

- En el **Capítulo 6**, "Histórico del proyecto", se describen las distintas fases de desarrollo del proyecto explicando las tareas llevadas a cabo, problemas surgidos y resultados, así como un resumen de este.

- Por último, en el **Capítulo 7**, “Conclusiones y líneas futuras”, se resumen las principales conclusiones extraídas de la realización del proyecto así como las líneas futuras de desarrollo.

Además de estos siete capítulos, al final del documento se incluyen cuatro anexos que completan aspectos relacionados con la resolución del proyecto:

- En el **Anexo A**, “Presupuesto”, se detalla el presupuesto del proyecto.

- En el **Anexo B**, “Entorno de trabajo”, se encuentran explicadas las herramientas *hardware* y *software* utilizadas para el sistema.

- En el **Anexo C**, “Manual de usuario”, se detalla a modo de guía para el usuario el uso de la aplicación

- En el **Anexo D**, "Resumen Extendido", se encuentra un resumen de la presente memoria en idioma Inglés.

- Para finalizar, en el **Anexo E**, “Glosario”, se incluye un glosario de términos.

Capítulo 2

Estado del arte

En este segundo capítulo, en primer lugar se realiza una introducción a los certificados digitales y más concretamente al estándar *X.509*, así como del uso y funcionalidades de estos. De igual manera, resulta imprescindible resaltar cuales son las necesidades que motivan la validación de dichos certificados enmarcados en escenarios diarios. Finalmente, para poder ofrecer al lector un esbozo más completo de lo que el proyecto representa, se explica el proceso necesario para llevar a cabo la validación de dichos certificados así como una descripción general de los distintos protocolos destinados a ello.

En el segundo punto, siguiendo en la línea de lo anteriormente expuesto se procede a explicar unos de los protocolos mencionados en el punto anterior, el protocolo SCVP, siendo este es el objeto de estudio del proyecto.

Una vez mencionados los procesos necesarios para llevar a cabo el diseño del sistema, los sucesivos puntos se encuentran dedicados a explicar cuáles son las tecnologías utilizadas para el desarrollo del proyecto. Inicialmente, se introduce el funcionamiento global de todo el sistema basado en un sistema distribuido de arquitectura cliente-servidor y posteriormente se presenta la plataforma Android, sistema operativo sobre el que se desarrolla el mismo, así como la API *Bouncy Castle* utilizada en el proyecto.

2.1 Validación de certificados digitales

En esta sección se introducen nociones básicas de criptografía y más concretamente de certificados digitales que permiten obtener una base para entender el desarrollo del proyecto.

2.1.1 Criptografía de clave asimétrica

La criptografía de clave asimétrica o criptografía de clave pública fue inventada en 1976 por los matemáticos Whit Diffie y Martin Hellman y constituye la base de la criptografía moderna, ya que permite servicios de seguridad importantes para las transacciones online tales como el cifrado de datos o la firma digital.

Esta se basa en la utilización de dos claves, una pública y otra privada, para el cifrado y descifrado de información. El fundamento de este tipo de criptografía se apoya en lo siguiente: lo que se encuentra cifrado con una clave privada necesita su correspondiente clave pública para ser descifrado. Y viceversa, lo cifrado con una clave pública sólo puede ser descifrado con su clave privada (Fuente [20]).

Por lo tanto, la clave privada únicamente debe ser conocida por su propietario mientras que la clave pública puede ser dada a conocer abiertamente. Así, si se desea enviar información confidencial a un usuario, esta deberá ser cifrada con su clave pública, de tal forma que solo ese usuario pueda descifrarla gracias a su clave privada.

En la siguiente **Figura 1** se encuentra ilustrado un escenario basado en el anteriormente mencionado cifrado de clave asimétrica.

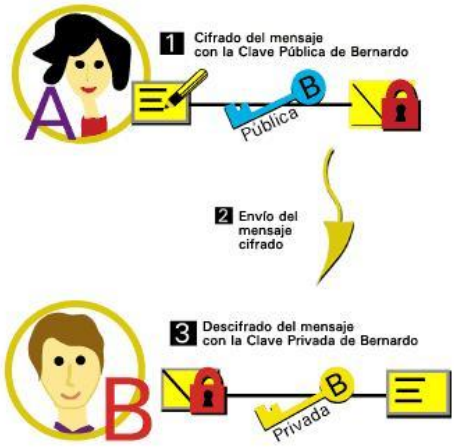

Figura 1. Cifrado de clave asimétrica (Fuente [1])

Bajo escenario, la autenticidad de la clave pública y su correspondencia con el usuario que la posee cobran gran relevancia a la hora del intercambio de información. Así pues, para la distribución de dicha clave y poder asegurar su autenticidad se hace uso de los certificados digitales.

2.1.2 Certificados digitales

Un certificado digital o también denominado certificado electrónico, es un documento digital generado por un tercero de confianza o entidad de certificación, que garantiza la correspondencia entre la clave pública del sujeto (persona física, organismo o empresa) y los datos de identidad de este.

Los datos que, normalmente contiene un certificado digital son los siguientes:

- Identificación del titular o entidad certificada
- Distintivos del certificado como numero de serie, fecha de expiración, fecha de emisión etc.
- La clave pública del titular
- Firma digital de la autoridad emisora del certificado

Existen diversos formatos para certificados digitales, sin embargo el más utilizado es el estándar *ITU-T X.509* cuya estructura se detalla a continuación en la **Figura 2** y se encuentra definida en la RFC 5280 [2].

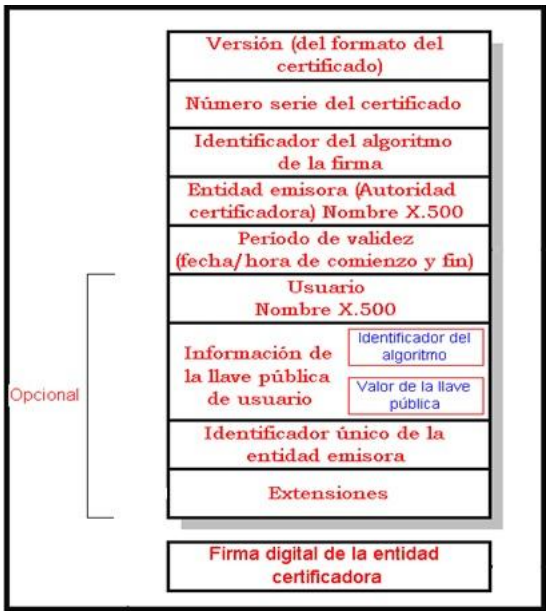

Figura 2. Formato X.509v3 de Certificado Digital (Fuente [3])

Las garantías que un certificado digital ofrece a un usuario son las de autenticidad de las personas y entidades que intervienen en el intercambio de información, confidencialidad de la información entre emisor y receptor en la comunicación, integridad de la información intercambiada y no repudio, que garantiza al titular del certificado la generación innegable de una firma vinculada a su certificado y le imposibilita refutar su titularidad en los mensajes firmados.

El cometido de un certificado digital es el de por un lado, autenticar la identidad del usuario ante terceros, firmar electrónicamente de forma que se garantice la integridad

tanto de los datos trasmitidos como de su procedencia y por último, cifrar datos de tal manera que sólo el destinatario del documento pueda acceder a su contenido.

Un claro ejemplo de aplicación del uso de los certificados digitales estaría representado por la firma de contratos a través de Internet en un entorno internacional. Un particular o empresa de cualquier país, puede firmar y enviar instantáneamente un contrato a otra empresa o particular para que firme dicho contrato y formalicen digitalmente una vinculación con plena validez jurídica. Otro ejemplo sería una transacción bancaria que requiera de la autenticación del usuario o cualquier trámite con la Administración Pública como puede ser la Agencia Tributaria. Estos ejemplos diarios de los usuarios que requieren de la utilización de certificados digitales válidos ponen de manifiesto importantes ventajas derivadas de su utilización como un importante ahorro de tiempo evitando desplazamientos para gestiones, ahorro en costes al permitir las gestiones telemáticas de trámites sin tener que imprimir documentación o poder realizar trámites *online*, a cualquier hora, desde cualquier lugar y de una forma segura.

2.1.3 Proceso de construcción y validación

Una vez situado en contexto el certificado digital y sus principales características se introduce el proceso de validación de este, el cual debe seguir la ejecución de dos subprocesos.

En primer lugar, definido un camino o ruta de certificación como una cadena de certificados en la que el emisor del primer certificado es un punto de confianza y el sujeto del último certificado es la entidad final que se intenta validar, la primera acción a llevar a cabo en el proceso de validación pasa por la construcción de dicha ruta de certificación entre el certificado en cuestión y la raíz de confianza. A esto se lo denomina construcción o descubrimiento de la ruta.

En segundo lugar, una vez construida dicha ruta se debe comprobar la validez de todos y cada uno de los certificados pertenecientes a esta. A este segundo proceso se lo denomina validación de la ruta. A la hora de proceder a la comprobación de la validez de un certificado digital se deben realizar una serie de procedimientos por los cuales se podrán obtener diferentes resultados sobre el mismo.

Un proceso de validación satisfactorio dará como resultado un certificado digital válido, el cual estará emitido por una CA (*Certification Authority*) que obedece a políticas de confianza definidas y además se encuentra dentro de la fecha de validez y no ha sido revocado. Por el contrario, si el proceso de validación de un certificado es insatisfactorio, el resultado será el de un certificado digital inválido o la posibilidad de que la validación se encuentre incompleta por un error. Un certificado digital inválido, se trata de un certificado que ha sido emitido por una *CA* en la que no se confía, o bien se encuentra expirado o revocado.

En definitiva, el procedimiento de validación incluye desempeñar, entre otras, las siguientes actuaciones:

- En primer lugar será necesario validar todos los certificados pertenecientes a la ruta de certificación, no solo el certificado sugerido.

- También será necesario comprobar el nivel de confianza de las distintas *CAs* usadas para la comprobación del estado del certificado.

- Asimismo, será necesario analizar los datos que se incluyen en los certificados.

- Y por último, comprobar los datos completos sobre la identidad del propietario del certificado digital (nombre, apellidos, organización, tipo de persona, NIF/CIF/VAT, etc.).

2.1.4 Protocolos

Cuando surgieron las PKIs (*Public Key Infrastructure*) o, en castellano Infraestructuras de Clave Pública, pocas revisaban el estado de los certificados.

Poco a poco se fueron implementando las CRLs o también llamadas, Listas de Revocación de certificados [2], como un primer mecanismo utilizado para probar la validez de los certificados digitales. Estas listas contienen los números de serie de aquellos certificados que hayan sido revocados por una CA concreta, siendo un certificado revocado aquel que ha sido cancelado o cuya validez ha sido anulada antes de que este expire.

Por lo tanto, con la utilización de dichas listas, cuando un tercero desea comprobar la validez de un certificado digital descarga una CRL actualizada desde los servidores de la misma CA que emitió dicho certificado. A continuación comprueba la autenticidad de la lista gracias a la firma digital de la CA y posteriormente comprueba que el número de serie del certificado en cuestión no se encuentra en la CRL.

Sin embargo, dichas listas cuentan con la única ventaja de poder ser consultadas sin necesidad de una conexión de datos permanente con cada CA, pero presentan numerosos inconvenientes como la posibilidad de que un certificado haya sido revocado pero no aparezca en la lista debido a una falta de actualización o un tamaño ineficiente de las listas para su tratamiento.

Como alternativa a este mecanismo basado en CRLs se presenta el protocolo OCSP (*Online Certificate Status Protocol*), un protocolo de consulta en línea que aporta información al momento sobre cada certificado en concreto y que se encuentra descrito en la RFC 2560 [4].

El funcionamiento de dicho protocolo se basa en una petición a un servidor formada por la versión del protocolo y los identificadores de los certificados que se desean validar. Estos identificadores están formados por el número de serie, el hash del DN (*Distinguished Name*) del emisor del certificado y el hash de la clave pública del mismo. La respuesta del servidor OCSP puede devolver una respuesta firmada lo cual indicaría que el certificado es válido, revocado o desconocido pero también puede devolver un código de error en cuyo caso la respuesta no tendría que estar firmada.

OCSP fue creado para solventar algunas deficiencias de las CRLs y por ende es preferible utilizar este protocolo para la validación de los certificados digitales ya que por ejemplo, puede proporcionar una información más adecuada y reciente del estado de

revocación de un certificado o eliminar la necesidad de que los clientes tengan que obtener y procesar las CRLs.

Sin embargo, OCSP proporciona solo un subconjunto de las funcionalidades que implementa el protocolo SCVP, el cual es el objeto de estudio de este proyecto ya que no existe ninguna implementación no comercial del mismo y que se encuentra explicado de una manera más pormenorizada en el siguiente apartado.

2.2 El protocolo SCVP

2.2.1 Introducción

Este protocolo se encuentra definido en la RFC 5055 [5], en la cual la información contenida en los mensajes se encuentra codificada en notación *ASN.1*.

SCVP es un protocolo diseñado para la realización de dos tareas, el DPD (*Delegated Path Discovery*) o descubrimiento de la ruta de certificación entre una raíz de confianza y un certificado digital *X.509* y el DPV (*Delegated Path Validation*) o validación de dicha ruta de certificación de acuerdo a una política definida.

Dicho protocolo está basado en la necesidad de un usuario de comprobar la validez de un certificado digital y su emisión por una CA de confianza al recibir dicho certificado. Además dicha cadena de validación puede presentar varias CA emisoras y por lo tanto sería necesario corroborar la firma digital de todas ellas trazando una ruta hacia atrás.

Una de las tareas para las que el protocolo SCVP ha sido definido, es el DPD, acción requerida por aquellas aplicaciones que pueden realizar la validación de la ruta de certificación, pero carecen de un método fiable y eficiente para construir o descubrir dicha ruta de certificación con validez. Por lo tanto, este tipo de clientes delegan el descubrimiento de la ruta al servidor SCVP pero no así la validación de este.

La otra tarea para la que el protocolo SCVP ha sido definido, es el DPV, adecuado para aquellas aplicaciones que requieren dos acciones sobre el certificado. La primera de ellas es la confirmación de que la clave pública pertenece a la identidad nombrada en el certificado y la segunda es la confirmación de que la clave pública se puede utilizar para la finalidad prevista. Este tipo de clientes delegan totalmente la construcción o descubrimiento de la ruta de certificación y la posterior validación en el servidor.

Así pues, el servidor SCVP da respuesta a dos tipos de peticiones. Por un lado atiende a aquellas peticiones que provengan de clientes que solo desean la construcción de una ruta fiable de certificación ya que la validación corre de su cuenta. Y por otro lado, atiende a aquellas peticiones que provengan de clientes que además de requerir la construcción de la ruta de certificación solicitan adicionalmente la validación completa de dicho certificado.

2.2.2 ASN.1

En esta sección se realizar una breve exposición sobre ASN.1 (*Abstract Syntax Notation One*) ya que esta es la notación utilizada para la definición de los datos en el protocolo SCVP y cuyo detalle se encuentra especificado en [6].

ASN.1 es una norma utilizada para representar datos independientemente de la máquina que se esté utilizando y sus formas de representación internas. La norma fue desarrollada como parte de la capa 6 de presentación del modelo de referencia OSI (*Open System Interconnection*) y publicada en la recomendación ITU-T X.208 | ISO/IEC 8824 en diciembre de 1987.

Existen distintos tipos de datos en esta notación cuya principal clasificación se basa según si son datos simples o también llamados primitivos, datos compuestos por estar construidos a partir de otros: simples o compuestos y, por último, datos predefinidos.

Los **tipos primitivos** son escalares, es decir, variables que almacenan un único valor y entre los que destacan los siguientes:

- `INTEGER` para representar números enteros bien positivos o negativos
- `REAL` para representar números reales.
- `BIT STRING` para representar cadenas de bits.
- `OCTET STRING` para representa cadenas de bytes.
- `OBJECT IDENTIFIER` para representar los identificadores de los objetos, es decir, la posición de un objeto dentro del árbol MIB (*Management Information Base*).
- `BOOLEAN` para representar valor lógico verdadero o falso.
- `NULL` para representar la ausencia de valor.
- `ENUMERATED` se utiliza para una enumeración de identificadores.

Los **tipos construidos** o tipos compuestos se usan para la creación tanto de arrays como de tablas, siendo los más importantes:

- `SEQUENCE` es una lista ordenada de tipos de datos diferentes. Es el tipo que se usa para almacenar una fila de una tabla.
- `SEQUENCE OF` es una lista ordenada de tipos de datos iguales. Es el tipo usado en tablas para almacenar todas las filas.
- `SET` es equivalente al `SEQUENCE`, pero la lista no está ordenada y todos los componentes de la lista SET deben ser diferentes sino la definición sería ambigua.

- `SET OF` es equivalente al `SEQUENCE OF` pero la lista no está ordenada.

- `CHOICE` es un tipo de datos en el que hay que elegir uno de entre los tipos disponibles en una lista.

Y por último se encuentran los **tipos de datos predefinidos** que son tipos de datos derivados de los anteriores pero con un nombre más descriptivo y entre los que se encuentran:

- `IpAddress` es un tipo que sirve para almacenar direcciones IPv4. Son 4 bytes y se define como `OCTET STRING (SIZE (4))`.

- `Counter` representa un contador que únicamente puede incrementar su valor y que cuando llega a su valor máximo, vuelve a cero.

- `Gauge` es un indicador de nivel cuyo valor se puede incrementar o decrementar.

- `IA5String` representa una cadena de texto en el alfabeto *IA5*.

- `NumericString` representa una cadena en el alfabeto 0-9 incluyendo el carácter espacio.

ASN.1 no define cómo se han de codificar los datos, sino que es una sintaxis abstracta para indicar el significado de los datos. Para la codificación de los datos se usan normas de codificación como son: BER, CER, DER , PER y XER.

En la siguiente sección se encuentran detalladas las normas de codificación tanto BER (*Basic Encoding Rules*) como DER *(Distinguised Encoding Rules)*, dado que estas son las dos únicas utilizadas para la definición del protocolo SCVP y por ende utilizadas a lo largo del proyecto.

2.2.3 Codificación BER y DER

Las reglas de codificación básicas, denominadas sintaxis de transferencia, en el contexto de ASN.1 especifican las secuencias de octetos exactas para codificar información abstracta en un flujo de bits único. Esto es, que pueda ser interpretado en cualquier máquina de la misma manera. La sintaxis BER, junto con dos subconjuntos de BER: CER *(Canonical Encoding Rules)* y DER, están definidas por el documento de estándares X.690 de la ITU-T, el cual es parte de las series de documentos ASN.1.

El formato BER especifica un formato auto-descriptivo para codificar las estructuras de datos ASN.1. Cada elemento de datos está codificado usando la codificación "tipo-longitud-valor". Es decir, por un identificador de tipos, una descripción de longitud, los elementos de datos actuales, y donde sea necesario, un marcador de fin de contenido. Este formato permite a un receptor decodificar la información ASN.1 desde una corriente incompleta, sin necesidad de tener un conocimiento previo del tamaño, contenido, o significado semántico de los datos.

Sin embargo, DER es una variante restringida de BER para la producción de sintaxis de transferencia inequívoca para estructuras de datos descritas por ASN.1 y además está considerada una forma canónica de BER.

Por lo tanto, DER es un subconjunto de BER para proporcionar exactamente una única forma de codificar un valor ASN.1. Este tipo de codificación cobra especial relevancia en situaciones en las que se necesita una codificación única, como por ejemplo en criptografía, ya que asegura que una estructura de datos que necesita ser firmada digitalmente produzca una representación serializada única.

La diferencia clave entre el formato BER y el formato DER estriba en la flexibilidad suministrada por las reglas de codificación básicas.

2.2.4 Funcionamiento SCVP

SCVP utiliza un modelo de petición-respuesta simple. Es decir, el cliente crea una solicitud que envía al servidor SCVP a la cual este debe responder creando una única respuesta. El uso típico de SCVP se espera que se realice a través de HTTP (*HiperText Transfer Protocol*), pero también se puede utilizar tanto en correo electrónico como en cualquier otro protocolo que pueda transportar objetos firmados digitalmente.

El protocolo se establece sobre dos de los mencionados pares petición-respuesta. El primero de ellos aborda la validación de certificados y el segundo se utiliza para determinar la lista de políticas de validación y los parámetros por defecto soportados por un servidor SCVP específico.

2.2.4.1 Política de validación

Una política de validación, según se define en la RFC 3379 [7], especifica las reglas y los parámetros que se utilizarán por el servidor SCVP para validar un certificado digital.

La petición de validación puede ser simple si se utiliza un OID (*Object Identifier*) para especificar el algoritmo a utilizar y todos los parámetros asociados a la política de validación. Pero la solicitud puede ser más compleja si dicha política de validación define algunos de los parámetros pero permite al cliente especificar otros de ellos. Cuando la política de validación define todos los parámetros necesarios, una solicitud SCVP sólo tiene que contener el certificado que deberá ser validado, la política de validación a la que se hace referencia, y los parámetros de tiempo de ejecución de la solicitud.

Un servidor publica las referencias de las políticas de validación que soporta y cuando estas políticas tienen parámetros que pueden ser anulados o cambiados manualmente, el servidor comunica igualmente los valores predeterminados de estos parámetros.

2.2.4.2 Algoritmos de validación

El algoritmo de validación se determina por acuerdo entre el cliente y el servidor y se representa mediante un OID. Dicho algoritmo define la comprobación que se debe llevar a cabo por el servidor para determinar si el certificado es válido.

El algoritmo de validación es uno de los parámetros de la política de validación correspondiente al servidor. SCVP define un algoritmo de validación por defecto que implementa el algoritmo básico de validación de rutas tal como se define en la RFC 5280 [2], y que permite al cliente solicitar información adicional sobre el certificado a validar. Se podrían especificar nuevos algoritmos de validación que definiesen verificaciones adicionales en caso de que fuera necesario.

2.2.4.3 Requerimientos de validación

Para que una ruta de certificación pueda ser considerada válida en base a una política de validación específica, esta debe ser válida tal y como se define en la RFC 5280 [2] , y se deben verificar todas las restricciones de la política de validación que se aplican a la ruta.

La comprobación de si un certificado se encuentra revocado es una característica opcional en la petición al servidor y la información de revocación de este puede ser distribuida en múltiples formatos. Los clientes deben especificar en las solicitudes si desean que se realice dicha comprobación y si la información de revocación debe ser devuelta en la respuesta.

Los servidores SCVP deben indicar las fuentes de información de revocación que son capaces de procesar, siendo estas:

- Respuestas generadas por el protocolo OCSP
- CRL completa que es una lista de revocación que contiene el registro de todos los certificados revocados de una CA.
- Delta CRL que es una lista de revocación que solo contiene los registros de los certificados revocados desde la última CRL que la CA generó.
- CRL indirecta que es una lista de revocación que contienen registros de certificados revocados por múltiples CAs.

2.2.4.4 Petición de validación

El cliente utiliza el elemento `CVRequest` para solicitar la validación de un certificado digital al servidor SCVP, y este se puede encontrar encapsulado en dos tipos de solicitud: protegida o no protegida.

El tipo de solicitud protegida se utiliza para autenticar al cliente ante el servidor o mantener la integridad de este como un cliente anónimo en el par de petición-respuesta. La protección es proporcionada por una firma digital o por un mensaje de código de

autenticación digital (MAC). Además el servidor SCVP puede requerir que todas las peticiones del cliente sean protegidas y por lo tanto descartar aquellas que no lo sean.

Por un lado, la sintaxis correspondiente al mensaje cuya solicitud es no protegida, que consiste en el valor binario correspondiente a la codificación DER del elemento `CVRequest` y que se encuentra encapsulado en un tipo de dato `ContentInfo` se estructurada de la siguiente manera:

```
ContentInfo {
  contentType   id-ct-scvp-certValRequest,
                          --(1.2.840.113549.1.9.16.1.10)
  Content       CVRequest }
```

Por otro lado, la sintaxis correspondiente al mensaje cuya solicitud está protegida, que consta del elemento `CVRequest` encapsulado en un elemento `SignedData` o `AuthenticatedData` y que está a su vez encapsulado en un `ContentInfo`, se encuentra estructurado como se detalla a continuación:

```
ContentInfo {
  contentType   id-signedData,
                          --(1.2.840.113549.1.7.2)
  Content       SignedData }
```

Si el cliente utiliza `SignedData` debe tener una clave pública asociada a su identidad y un certificado adecuado para la firma de la solicitud. Debe incluir su propio certificado digital, utilizado para la firma, e incluir exactamente un `SignerInfo` en los campos correspondientes para ello del elemento `SignedData`, cuya sintaxis es la siguiente:

```
SignedData {
  version               CMSVersion,
  digestAlgortihms      DigestAlgorithmIdentifiers,
  encapContentInfo      EncapsulatedContentInfo,
  certificates          CertificateSet,
  crls                  CertificateRevocationLists,
  signedInfos           SET OF SignerInfos }
```

A su vez, `EncapsulatedContentInfo` está formado por un elemento `eContentType` cuyo valor debe ser `id-ct-scvp-certValRequest` y un campo `eContent` con el valor binario correspondiente a la codificación DER del elemento `CVRequest` de la siguiente manera:

```
EncapsulatedContentInfo {
  eContentType   id-ct-scvp-certValRequest,
                          --(1.2.840.113549.1.9.16.1.10)
  eContent       CVRequest }
```

En `CVRequest`, la estructura es la siguiente, siendo solamente obligatorios tanto `cvRequestVersion` como el elemento `query`.

```
CVRequest ::= SEQUENCE {
  cvRequestVersion        INTEGER DEFAULT 1,
```

```
    query                    Query,
    requestorRef        [0]  GeneralNames OPTIONAL,
    requestNonce        [1]  OCTET STRING OPTIONAL,
    requestorName       [2]  GeneralName OPTIONAL,
    responderName       [3]  GeneralName OPTIONAL,
    requestExtensions   [4]  Extensions OPTIONAL,
    signatureAlg        [5]  AlgorithmIdentifier OPTIONAL,
    hashAlg             [6]  OBJECT IDENTIFIER OPTIONAL,
    requestorText       [7]  UTF8String (SIZE (1..256)) OPTIONAL}
```

El elemento `Query` especifica uno o varios certificados los cuales son el objeto de la consulta. Este debe contener al menos un elemento `queriedCerts`, un elemento `checks` y un elemento `validationPolicy`, cuya estructura completa es la siguiente:

```
Query ::= SEQUENCE {
    queriedCerts            CertReferences,
    checks                  CertChecks,
    wantBack            [1] WantBack OPTIONAL,
    validationPolicy        ValidationPolicy,
    responseFlags           ResponseFlags OPTIONAL,
    serverContextInfo   [2] OCTET STRING OPTIONAL,
    validationTime      [3] GeneralizedTime OPTIONAL,
    intermediateCerts   [4] CertBundle OPTIONAL,
    revInfos            [5] RevocationInfos OPTIONAL,
    producedAt          [6] GeneralizedTime OPTIONAL,
    queryExtensions     [7] Extensions OPTIONAL }
```

La lista de `Certificate Reference` del elemento `queriedCerts` le indica al servidor, el o los certificados de los que el cliente desea la información. `Checks` indica la comprobación que se debe realizar sobre el o los certificados en cuestión y `validationPolicy` indica la política de validación que el cliente desea que el servidor utilice.

2.2.4.5 Respuesta de validación

El servidor SCVP utiliza el elemento `CVResponse` en respuesta a un `CVRequest` de validación.

A su vez, dicho mensaje de respuesta SCVP puede tener distintos formatos y en función de ese formato el mensaje se encontrará encapsulado en un tipo de respuesta protegida o no protegida.

- Una respuesta de éxito a una solicitud cuyo campo `protectResponse` tiene valor `FALSE` **no debería** ser protegida por el servidor.

- El servidor **debe** proteger todas las demás respuestas de éxito y en caso de que no fuera capaz de devolver una respuesta protegida de éxito debido a la política local, entonces debe devolver una respuesta de error.

- Una respuesta de error, aunque sea una solicitud realizada sobre transporte seguro como TLS (*Transfer Layer Security*), **no debería** ser protegida por el servidor.

- Una respuesta de error a una solicitud cuyo campo `protectResponse` tiene valor `FALSE` **no debería** ser protegida por el servidor.

- Una respuesta de error a una solicitud autenticada **debería** ser protegida por el servidor.

- Una respuesta de error a una solicitud `AuthenticatedData` cuyo valor MAC es válido **debe** ser protegida por el servidor.

- Todas las demás tipos de respuestas **no deben** ser protegidas por el servidor.

Las respuestas de éxito son enviadas por el servidor cuando haya cumplido satisfactoriamente con los requisitos de la solicitud, es decir, este haya sido capaz de construir correctamente la ruta de certificación en base a la política de validación y/o la validación correspondiente.

Las respuestas de error son enviadas por el servidor si es incapaz de realizar validaciones con ayuda de la política de validación requerida o la solicitud contiene una opción no compatible.

Para peticiones y respuestas protegidas, el servidor SCVP debe soportar `SignedData` y debería soportar también `AuthenticatedData`. Si el servidor realiza una respuesta protegida a una solicitud protegida debe utilizar el mismo mecanismo de protección (`SignedData` o `AuthenticatedData`) que en la solicitud.

Por un lado, la sintaxis correspondiente al mensaje cuya respuesta es no protegida, que consiste en el valor binario correspondiente a la codificación BER del elemento `CVResponse` y que se encuentra encapsulado en un tipo de dato `ContentInfo` se estructurada de la siguiente manera:

```
ContentInfo {
        contentType         id-ct-scvp-certValResponse,
                               -- (1.2.840.113549.1.9.16.1.11)
        content             CVResponse }
```

Por otro lado, la sintaxis correspondiente al mensaje cuya respuesta está protegida, que consta del elemento `CVResponse` encapsulado en un elemento `SignedData` o `AuthenticatedData` y que está a su vez encapsulado en un `ContentInfo`, se encuentra estructurado como se detalla a continuación:

```
ContentInfo {

  contentType   id-signedData,
                              --(1.2.840.113549.1.7.2)
  Content       SignedData }
```

El servidor SCVP debe incluir su propio certificado digital, utilizado para la firma, e incluir exactamente un `SignerInfo` en los campos correspondientes a ellos del elemento `SignedData,` cuya sintaxis es la siguiente:

```
SignedData {
  version               CMSVersion,
  digestAlgortihms      DigestAlgorithmIdentifiers,
  encapContentInfo      EncapsulatedContentInfo,
  certificates          CertificateSet,
  crls                  CertificateRevocationLists,
  signedInfos           SET OF SignerInfos }
```

A su vez, `EncapsulatedContentInfo` está formado por un elemento `eContentType` cuyo valor debe ser `id-ct-scvp-certValResponse` y un campo `eContent` con el valor binario correspondiente a la codificación DER del elemento `CVResponse` de la siguiente manera:

```
EncapsulatedContentInfo {
  eContentType   id-ct-scvp-certValResponse,
                          --(1.2.840.113549.1.9.16.1.11)
  eContent       CVResponse }
```

En `CVResponse`, la estructura es la siguiente, siendo únicamente obligatorios los campos `cvResponseVersion`, `serverConfigurationID`, `producedAt` y `responseStatus`.

```
CVResponse ::= SEQUENCE {
  cvResponseVersion         INTEGER,
  serverConfigurationID     INTEGER,
  producedAt                GeneralizedTime,
  responseStatus            ResponseStatus,
  respValidationPolicy  [0] RespValidationPolicy OPTIONAL,
  requestRef            [1] RequestReference OPTIONAL,
  requestorRef          [2] GeneralNames OPTIONAL,
  requestorName         [3] GeneralNames OPTIONAL,
  replyObjects          [4] ReplyObjects OPTIONAL,
  respNonce             [5] OCTET STRING OPTIONAL,
  serverContextInfo     [6] OCTET STRING OPTIONAL,
  cvResponseExtensions  [7] Extensions OPTIONAL,
  requestorText          [8] UTF8String (SIZE (1..256))
                             OPTIONAL }
```

Siendo la estructura del elemento `ResponseStatus` la siguiente:

```
ResponseStatus ::= SEQUENCE {
        statusCode            CVStatusCode DEFAULT  okay,
        errorMessage          UTF8String OPTIONAL }
```

```
CVStatusCode ::= ENUMERATED {
        okay                                (0),
        skipUnrecognizedItems               (1),
        tooBusy                            (10),
        invalidRequest                     (11),
        internalError                      (12),
        badStructure                       (20),
        unsupportedVersion                 (21),
        abortUnrecognizedItems             (22),
        unrecognizedSigKey                 (23),
```

```
            badSignatureOrMAC                   (24),
            unableToDecode                      (25),
            notAuthorized                       (26),
            unsupportedChecks                   (27),
            unsupportedWantBacks                (28),
            unsupportedSignatureOrMAC           (29),
            invalidSignatureOrMAC               (30),
            protectedResponseUnsupported        (31),
            unrecognizedResponderName           (32),
            relayingLoop                        (40),
            unrecognizedValPol                  (50),
            unrecognizedValAlg                  (51),
            fullRequestInResponseUnsupported    (52),
            fullPolResponseUnsupported          (53),
            inhibitPolicyMappingUnsupported     (54),
            requireExplicitPolicyUnsupported    (55),
            inhibitAnyPolicyUnsupported         (56),
            validationTimeUnsupported           (57),
            unrecognizedCritQueryExt            (63),
            unrecognizedCritRequestExt          (64) }
```

2.2.4.6 Petición de política de validación

El cliente utiliza el elemento `ValPolRequest` para solicitar información acerca de la política del servidor SCVP e información de configuración, incluida en la lista de políticas de validación que admite el servidor.

La sintaxis del mensaje de solicitud, que consiste en el valor binario correspondiente a la codificación BER del elemento `ValPolRequest` y que se encuentra encapsulado en un tipo de dato `ContentInfo` se detalla a continuación de la siguiente manera:

```
ContentInfo {
  contentType   id-ct-scvp-valPolRequest,
                          --(1.2.840.113549.1.9.16.1.12)
  Content       ValPolRequest }
```

Estando a su vez el elemento `ValPolRequest` estructurado de la siguiente forma:

```
ValPolRequest ::= SEQUENCE {
  vpRequestVersion           INTEGER DEFAULT 1,
  requestNonce               OCTET STRING }
```

2.2.4.7 Respuesta de política de validación

El servidor SCVP utiliza el elemento `ValPolResponse` en respuesta a un `ValPolRequest` para enviar al cliente la información acerca de su política de validación. Este además deberá firmar dicha respuesta utilizando su certificado digital.

La sintaxis de dicho mensaje de respuesta, que consta del elemento `ValPolResponse` encapsulado en un elemento `SignedData` y que está a su vez encapsulado en un `ContentInfo`, se estructura de la siguiente forma:

```
ContentInfo {
  contentType   id-signedData,
                          --(1.2.840.113549.1.7.2)
  Content       SignedData }
```

El servidor SCVP debe incluir su propio certificado digital, utilizado para la firma, e incluir exactamente un `SignerInfo` en los campos correspondientes a ellos del elemento `SignedData`, cuya sintaxis es la siguiente:

```
SignedData {
  version               CMSVersion,
  digestAlgortihms      DigestAlgorithmIdentifiers,
  encapContentInfo      EncapsulatedContentInfo,
  certificates          CertificateSet,
  crls                  CertificateRevocationLists,
  signedInfos           SET OF SignerInfos }
```

A su vez, `EncapsulatedContentInfo` está formado por un elemento `eContentType` cuyo valor debe ser `id-ct-SCVP-valPolResponse` y un campo `eContent` con el valor binario correspondiente a la codificación DER del elemento `ValPolResponse` de la siguiente manera:

```
EncapsulatedContentInfo {
  eContentType   id-ct-scvp-valPolResponse,
                          --(1.2.840.113549.1.9.16.1.13)
  eContent       ValPolResponse }
```

Estando `ValPolResponse`, el elemento principal de la respuesta al cliente, estructurado de la siguiente manera:

```
ValPolResponse ::= SEQUENCE {
  vpResponseVersion          INTEGER,
  maxCVRequestVersion        INTEGER,
  maxVPRequestVersion        INTEGER,
  serverConfigurationID      INTEGER,
  thisUpdate                 GeneralizedTime,
  nextUpdate                 GeneralizedTime OPTIONAL,
  supportedChecks            CertChecks,
  supportedWantBacks         WantBack,
  validationPolicies         SEQUENCE OF OBJECT IDENTIFIER,
  validationAlgs             SEQUENCE OF OBJECT IDENTIFIER,
  authPolicies               SEQUENCE OF AuthPolicy,
  responseTypes              ResponseTypes,
  defaultPolicyValues        RespValidationPolicy,
  revocationInfoTypes        RevocationInfoTypes,
  signatureGeneration        SEQUENCE OF AlgorithmIdentifier,
  signatureVerification      SEQUENCE OF AlgorithmIdentifier,
  hashAlgorithms             SEQUENCE SIZE (1..MAX) OF
                                        OBJECT IDENTIFIER,
  serverPublicKeys           SEQUENCE OF KeyAgreePublicKey
                                        OPTIONAL,
  clockSkew                 INTEGER DEFAULT 10,
  requestNonce               OCTET STRING OPTIONAL }
```

2.3 Sistema distribuido

En esta sección se realiza una pequeña introducción al tipo de sistema distribuido empleado para el diseño del proyecto así como al protocolo de aplicación sobre el que este se sustenta, HTTP.

Por un lado, un sistema distribuido se define como una colección de ordenadores autónomos conectados por una red con el software distribuido adecuadamente para que el sistema sea percibido por los usuarios como una única entidad capaz de proporcionar facilidades de computación. Es decir, un sistema distribuido se define como, aquel cuyos componentes *hardware* y *software* se comunican y coordinan sus acciones mediante el paso de mensajes para el logro de un objetivo.

Este tipo de sistema, es en resumen, la visión de sistema único a partir de un conjunto de elementos interconectados.

El modelo cliente-servidor de un sistema distribuido es el modelo más conocido y más ampliamente adoptado en la actualidad. En dicho modelo, a la máquina que solicita un determinado servicio se la denomina cliente y a la máquina que proporciona el servicio en cuestión se la denomina servidor. En la arquitectura de este modelo de aplicación distribuida las tareas se reparten entre los proveedores de recursos o servidores, y los demandantes o clientes. En definitiva, un cliente realiza peticiones al servidor, que le da respuesta.

Dado el formato que el protocolo SCVP establece para el intercambio de mensajes, la arquitectura estará basada en un diálogo entre el cliente, que en nuestro caso será un dispositivo con sistema operativo Android y un servidor SCVP.

Tal como se explica a lo largo de la sección anterior, el diálogo en base al protocolo SCVP se sustenta en el intercambio de cuatro mensajes entre el cliente y el servidor, dos de los cuales son utilizados para obtener información sobre la política de validación del servidor y los otros dos para la construcción y/o validación del camino de certificación requerida por el cliente. Por lo tanto, el diseño del sistema completo se establece bajo una arquitectura directa entre un servidor SCVP y un cliente que se conecta a él.

Por otro lado, el formato establecido para el intercambio de mensajes en el sistema se realiza sobre el protocolo HTTP que se encuentra detallado a continuación.

2.3.1 HTTP

El protocolo de transferencia de hipertexto HTTP, es un protocolo perteneciente a la capa de aplicación del modelo TCP/IP, usado para la transferencia de información entre sistemas de forma clara y rápida. Este protocolo ha sido utilizado por el World-Wide Web desde 1990, siendo actualmente la versión HTTP/1.1 la más reciente, cuya especificación se encuentra recogida en la RFC 2616 [8].

Desde el punto de vista de las comunicaciones, HTTP se establece sobre la capa de conexión de TCP/IP y funciona de la misma manera que el resto de los servicios comunes

del entorno UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (*Transmission Control Protocol)* y espera las solicitudes de conexión de los clientes.

HTTP permite usar una serie de métodos para indicar la finalidad de la petición y el diálogo con el servidor, destacando entre ellos el método GET, para recoger un objeto; POST para enviar información al servidor y HEAD para solicitar las características de un objeto.

Además se basa en otros conceptos y estándares como URI *(Uniform Resource Identifier*), URL (*Uniform Resource Location)* y URN (*Uniform Resource Name)*, para indicar el recurso al que hace referencia la petición.

El protocolo HTTP se basa en un paradigma de peticiones y respuestas de modo que:

- Un cliente envía una petición en forma de método, una URI, y una versión de protocolo seguida de los modificadores de la petición de forma parecida a un mensaje MIME (*MultiPurpose Mail Extensions*), información sobre el cliente y al final un posible contenido.

- El servidor contesta con una línea de estado que incluye la versión del protocolo y un código que indica éxito o error, seguido de la información del servidor en forma de mensaje MIME y un posible contenido.

El diálogo con el servidor HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo.

Sólo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta. La estructura general de los dos tipos de mensajes se puede ver en la siguiente **Figura 3.**

Mensaje de solicitud		**Mensaje de respuesta**
Comando HTTP + parámetros Cabeceras del requerimiento (línea en blanco) Información opcional		Resultado de la solicitud Cabeceras de la respuesta (línea en blanco) Información opcional

Figura 3. Estructura general para los dos tipos de mensajes HTTP (Fuente [9])

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que en la respuesta contiene el resultado de la operación y un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (obligatorias u opcionales), que condicionan y matizan el funcionamiento del protocolo. El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo.

Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Asimismo, HTTP cuenta con la característica de ser un protocolo

sin estado, es decir, cada petición de un cliente a un servidor no es influida por las transacciones anteriores.

A continuación, se detallan las convenciones estipuladas en las peticiones y respuestas de SCVP sobre HTTP, en cuanto a formato y transporte se refiere.

En líneas generales:

- Todos los clientes deben utilizar el método POST a la hora de realizar sus solicitudes.

- El servidor debe utilizar el código de respuesta 200 para aquellas que sean respuesta de éxito.

- Los clientes pueden intentar enviar peticiones seguras HTTPS utilizando TLS 1.0 o posterior, aunque los servidores no están obligados a soportar TLS.

A continuación, se detalla en particular el formato del mensaje HTTP para cada uno de los cuatro posibles mensajes solicitud-respuesta que se llevan a cabo en el protocolo SCVP.

1. Petición de validación:

   Mensaje de petición HTTP con el correspondiente método POST seguido de la cabecera `Content-Type` de valor `application/scvp-cv-request` y del cuerpo del mensaje con el valor binario de la codificación DER del `CVRequest`, encapsulado como se describe en la sección [2.4.4.4].

2. Respuesta de validación

   Mensaje de respuesta HTTP seguido de la cabecera `Content-Type` de valor `application/scvp-cv-response` y del valor binario de la codificación BER del `CVResponse`, encapsulado como se describe en la sección [2.4.4.5].

3. Petición política de validación

   Mensaje de petición HTTP con el correspondiente método POST seguido de la cabecera `Content-Type` de valor `application/scvp-vp-request` y del cuerpo del mensaje con el valor binario de la codificación BER del `ValPolRequest`, encapsulado como se describe en la sección [2.4.4.6].

4. Respuesta política de validación

   Mensaje de respuesta HTTP seguido de la cabecera `Content-Type` de valor `application/scvp-vp-response` y del valor binario de la codificación DER del `ValPolResponse`, encapsulado como se describe en la sección [2.4.4.7].

2.4 El sistema operativo Android

Por último para completar el capítulo dos, en esta sección se va a realizar una introducción al sistema operativo Android, utilizado para el diseño del cliente dentro del sistema, así como de la API criptográfica denominada *Bouncy Castle*.

Dado que la implementación realizada en el servidor utiliza el lenguaje Java y este está basado en las mismas funciones criptográficas descritas a continuación para Android, no se considera necesario agregar otra sección destinada a ello.

Android es un sistema operativo móvil, desarrollado inicialmente por la compañía Android Inc, en la actualidad propiedad de Google y supervisado por miembros de la OHA (*Open Handset Alliance*).

La presentación de la plataforma Android, se realizó el 5 de noviembre de 2007, junto con la mencionada fundación OHA, un consorcio de 48 compañías de *hardware*, *software* y de telecomunicaciones comprometidas con la promoción de estándares abiertos para móviles.

Este sistema basado en el núcleo de Linux, se trata de una plataforma completa de *software* para dispositivos móviles, que agrupa un sistema operativo, *middleware* y aplicaciones móviles.

Android, es además, una plataforma de código abierto lo que permite el desarrollo de aplicaciones por terceros. Se gestiona mediante las herramientas proporcionadas por Google tales como, el SDK (*Software Development Kit*) de Android, que provee a los programadores de las herramientas necesarias o APIs necesarias para el desarrollo de dichas aplicaciones mediante el uso del lenguaje de programación Java.

A continuación se presentan algunas de las características más destacadas del sistema:

- Android, es un sistema operativo pensado para optimizar recursos, lo cual tiene especial relevancia en terminales móviles donde resulta primordial no consumir demasiada batería. Para ello cuenta con su propia máquina virtual, denominada *Dalvik*, pensada y diseñada para realizar una gestión eficiente tanto de las aplicaciones como de memoria.

- Como ya se mencionó anteriormente, es una plataforma de código abierto, lo que facilita el desarrollo de aplicaciones sin necesidad del pago de regalías o *royalties*. Además, estas se encuentran implementadas en Java, lo que garantiza su ejecución en cualquier tipo de procesador avalando un gran nivel de portabilidad.

- Android, incluye además una base de datos *SQLite* propia, para el almacenamiento estructurado de datos. Al tratarse de una base de datos local, está limitada a información guardada en el propio dispositivo móvil pero aporta facilidad y sencillez a la hora de ser utilizada e integrada en las aplicaciones.

- Otro aspecto clave es que cuenta con *Webkit*, un motor de navegación de código abierto, al que se le permiten añadir más características.

- Su arquitectura, basa alguno de sus componentes en Internet, como los ficheros XML(*eXtensible Markup Language*), utilizados para el diseño de las interfaces de la aplicación y que facilitan la visión de la aplicación en pantallas.

- En cuanto a prestaciones multimedia se refiere, dado que Android se encuentra orientado a dispositivos móviles con aplicaciones multimedia, soporta diferentes formatos de audio, video e imágenes como son: MPEG4, MP3, 3GP, JPG, GIF.

- Además cuenta con el sistema de localización GPS, así como de Bluetooth, 3G, WiFi, cámara y reconocimiento y síntesis de voz entre otros.

2.4.1 Arquitectura Android

La arquitectura de Android está formada por cuatro capas de *software* donde cada una de ellas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.

Partiendo de la capa inferior y no siendo esta pública se encuentra el núcleo o Kernel, formado por un conjunto de drivers basados en Linux. Un nivel por encima se encuentra situado el Runtime de Android y las librerías nativas, que no son accesibles directamente sino a través de un nivel superior, el Framework o entorno de aplicación. Este junto con la última capa, la de aplicaciones, son totalmente públicas y se puede acceder a ellas libremente.

En la **Figura 4** se muestra la estructura global de las capas y sus principales componentes, las cuales se encuentran explicadas a continuación.

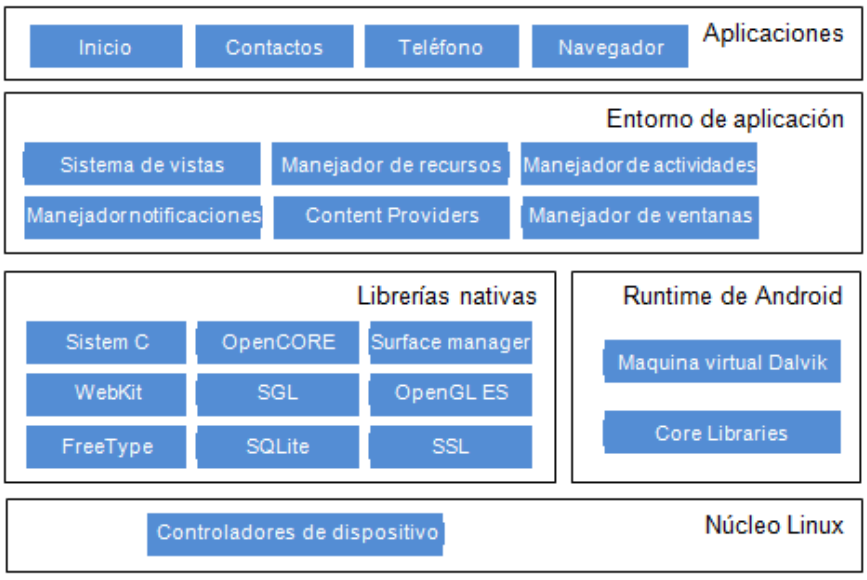

Figura 4. Arquitectura global de Android (Fuente [10])

2.4.1.1 Núcleo Linux

En la parte más inferior se encuentra el núcleo de Android, que basado en el sistema operativo Linux 2.6, es la base de la pila de software del sistema y por ello se encarga de las funciones más básicas tales como la gestión de los *drivers* de los dispositivos, seguridad, gestión de la memoria y la administración de procesos entre otras.

Esta es la capa encargada de proporcionar la abstracción necesaria entre los elementos *hardware* y el resto de las capas de *software* que pertenecen a la arquitectura global del sistema.

2.4.1.2 Librerías nativas y Runtime de Android

La siguiente capa se corresponde con las librerías nativas utilizadas por Android. Éstas han sido escritas utilizando C/C++ y su finalidad es la de facilitar funcionalidades a las aplicaciones mediante la utilización de código ya implementado.

Entre las librerías más destacadas se encuentran *System C Library* basada en la librería BSD de C estándar (*libc*) que incluye todas las cabeceras y funciones según el estándar del lenguaje C; *Surface Manager* encargada de componer los diferentes elementos de navegación de pantalla, además gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento; *SQLite* para la creación y gestión de bases de datos relacionales o *WebKit* encargada de proporcionar un motor para las aplicaciones de tipo navegador.

Al mismo nivel que las librerias de Android se sitúa el *Runtime* de Android, o también conocido como entorno de ejecución, formado por la máquina virtual Dalvik y las *Core Libraries*.

Por un lado, la máquina virtual, es el componente principal del *Runtime* y la encargada de la ejecución de las aplicaciones. Dicha máquina está basada en registros y ejecuta clases por el compilador de Java que hayan sido previamente transformadas al formato .dex (*Dalvik Executable*) para el ahorro de memoria. Y por otro lado dentro del *Runtime* de Android se encuentran las denominadas *Core Libraries* que recogen las librerías con la mayoría de las funcionalidades de Java.

2.4.1.3 Entorno de Aplicación

Esta es la capa encargada de proporcionar un conjunto de herramientas de desarrollo para cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "*framework",* representado por este nivel.

Una característica importante del entorno de aplicación es que se aprovecha el lenguaje Java para todo aquello que el SDK de Android no es capaz de ofrecer en su estándar del JRE (*Java Runtime Environment*).

Está pensado para la reutilización de componentes, es decir, una aplicación puede utilizar funcionalidades de otra creada anteriormente para su desarrollo.

Los servicios destacados en este nivel son el sistema de vistas, para la interfaz visual de las aplicaciones; el manejador de recursos, que proporciona el acceso a los recursos que no son código; el manejador de actividades, que gestiona el ciclo de vida de las aplicaciones en Android; el manejador de notificaciones, mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución; los proveedores de contenidos, que permite a cualquier aplicación compartir sus datos con las demás aplicaciones y el manejador de ventanas que gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.

2.4.1.4 Aplicaciones

Esta es la capa formada por las aplicaciones instaladas en el dispositivo, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

Entre las aplicaciones incluidas como base podemos encontrar un navegador Web, un cliente de email, un calendario, programa de SMS (*Short Message Service*), mapas, contactos, y algunos otros servicios mínimos. En este nivel se encuentra además la aplicación Inicio, la cual permite la ejecución de otras aplicaciones desde los diferentes escritorios y la ejecución de *widgets.*

Las aplicaciones son programas normalmente escritos en Java y desarrollados mediante la utilización del SDK de Android. Asimismo, cabe la posibilidad de que la programación se desarrolle en C o C++.

Además las aplicaciones pueden albergar cuatro tipos de componentes, que se encargan de definir el comportamiento y funcionalidad de la misma, y que son los siguientes:

- Por un lado, las actividades son las que comprenden las diferentes pantallas que un usuario ve en el dispositivo y permiten la interacción con él. Son en definitiva, tareas que se llevan a cabo en la aplicación y que tienen una interacción con el usuario. Existe una actividad principal, que es la primera en ejecutarse al lanzarse la aplicación y posteriormente, el movimiento entre pantallas se realiza llamando a nuevas actividades.

- Los servicios son, en cambio, procesos que se ejecutan en *background* sin necesidad de interacción con el usuario ni de despliegue de una pantalla y posiblemente por largos periodos de tiempo.

- Por otro lado, los receptores de anuncios son los encargados de la recepción y reacción frente a anuncios tanto si estos son originados por el sistema como si proceden de las aplicaciones. Se trata de un código que permanece inactivo hasta que se recibe el evento con el que se encuentra relacionado.

- Y por último, los proveedores de servicios son utilizados cuando los datos deben ser compartidos de unas aplicaciones a otras. Esta clase implementa un conjunto de métodos estándar para aquellas aplicaciones que comparten datos puedan almacenar y extraer la información que contiene el proveedor.

Las aplicaciones cuentan con un ciclo de vida, el cual recoge las diferentes etapas que pueden seguir sus procesos desde que esta comienza hasta que finaliza. En Android, es el sistema el que determina el ciclo de vida de una aplicación y no esta última. Es decir, el sistema determina su duración en base a las partes de la aplicación que estén ejecutando en ese momento, la importancia de estas para el usuario y la memoria disponible.

Android clasifica de manera jerárquica los procesos según la importancia que tengan para el sistema con el fin de poder establecer cuáles deben ser mantenidos en este.

Son cinco los niveles establecidos en la jerarquía de procesos y estos se encuentran representados de mayor a menor prioridad en la **Figura 5** y explicados a continuación.

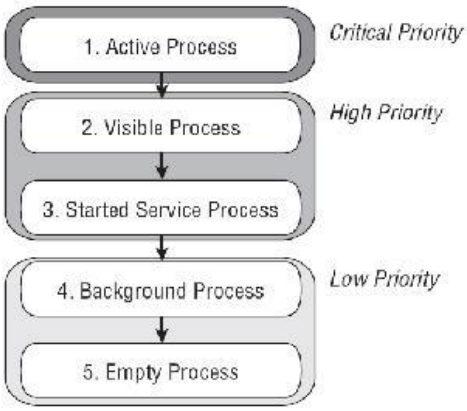

Figura 5. Jerarquía de procesos en Android (Fuente [11])

- Proceso en primer plano (*Active process*) es un proceso que aloja una Actividad en la pantalla y con la que el usuario está interactuando. Este tipo de procesos serán eliminados como último recurso si el sistema necesitase memoria.

- Proceso visible (*Visible process*) es un proceso que aloja una Actividad pero esta no se encuentra en primer plano. Esto ocurre en situaciones dónde la aplicación muestra un cuadro de diálogo para interactuar con el usuario. Este tipo de procesos no será eliminado en caso que sea necesaria la memoria para mantener a todos los procesos del primer plano corriendo.

- Proceso de servicio (*Started service process*) es un proceso que aloja un Servicio. Este tipo de procesos no son visibles y suelen ser importantes para el usuario.

- Proceso en segundo plano (*Background process*) es un proceso que aloja una Actividad que no es actualmente visible para el usuario. Normalmente la eliminación de estos procesos no suponen un gran impacto para la actividad del usuario. Es muy usual que existan numerosos procesos de este tipo en el sistema, por lo que el sistema mantiene una lista para asegurar que el último proceso visto por el usuario sea el último en eliminarse en caso de necesitar memoria.

- Proceso vacío (*Empty process*) es un proceso que no aloja ningún componente. La razón de existir de este proceso es tener una caché disponible de la aplicación para su próxima activación. Por lo tanto, son los primeros procesos en ser eliminados para equilibrar los recursos en memoria.

2.4.2 Bouncy Castle

Para concluir con este capítulo resulta imprescindible realizar una pequeña introducción a la librería o API *Bouncy Castle* sobre la que se sustenta el desarrollo criptográfico del proyecto, ya que esta supone una solución en Java para el desarrollo de infraestructura de clave pública y de la que se obtuvo información en la página principal de *Bouncy Castle* [12].

El paquete *Bouncy Castle* (BC) es una implementación Java de algoritmos criptográficos, organizado de forma que su API sea adaptable para su uso bajo cualquier entorno con la infraestructura adicional para ajustarse a los algoritmos del *framework* JCE (*Java Cryptography Extension*).

Bouncy Castle es una implementación de código abierto de JCE, que apoya, entre otras cosas, los certificados S/MIME, CMS, PKCS7, TEA, XTEA, SHA224 y X.509. Por lo tanto, contiene una API clave que no es dependiente de la JCE.

Entre las funcionalidades y características que ofrecen las APIs *Bouncy Castle* para Java destacan, para la realización del proyecto, las siguientes:

- Una API de criptografía ligera.
- Un proveedor de JCE (*Java Cryptography Extension*) y JCA (*Java Cryptography Architecture*).
- Una librería para la lectura y escritura codificada objetos ASN.1.
- API ligeros para TLS (RFC 2246, RFC 4346) y DTLS (RFC 4347).
- Generadores para las versiones 1 y 3 certificados *X.509*, versión 2 CRL y archivos PKCS12.
- Generadores para la versión 2 de certificados de atributos *X.509*
- Generadores / Procesadores de S/MIME y CMS (PKCS7 / RFC 3852).
- Generadores / Procesadores de OCSP (RFC 2560).
- Generadores / Procesadores de OpenPGP (RFC 4880).
- Generadores / Procesadores de Datos Validación y Certificación Server (DVCS) - RFC 3029.

Capítulo 3

Descripción general del sistema

En este capítulo 3 se pretende facilitar una visión general de la aplicación por medio de una descripción de la arquitectura global de la misma y de todos los elementos que la componen, así como de las relaciones que estos mantienen entre sí.

Para ello, se realiza también una exposición de las funcionalidades criptográficas implementadas por el sistema, así como del diseño en módulos realizado en base a su aportación al sistema. De igual manera, el capítulo cuenta con la especificación de los requisitos funcionales con los que cumple la aplicación y de los casos de uso de la misma.

Por lo tanto, el objetivo fundamental de este capítulo es clarificar el funcionamiento general del sistema a alto nivel y de este modo servir de introducción a los siguientes capítulos donde se analizan aspectos más técnicos del mismo.

3.1 Arquitectura

Para una primera aproximación a la arquitectura global del sistema implementado se definen los elementos involucrados en este, así como las interacciones existentes entre ellos. Dicha arquitectura se corresponde con el esquema mostrado en la **Figura 6**.

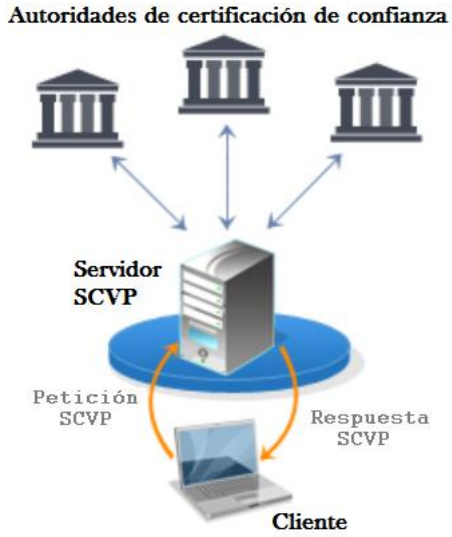

Figura 6. Arquitectura general del sistema (Fuente [13])

Tal y como se aprecia en dicha figura, el flujo de trabajo típico del sistema se establece en base a un **servidor SCVP** al que se conecta el usuario de la aplicación o **cliente.**

Por un lado el cliente final de la aplicación envía una solicitud de validación de certificados al servidor SCVP, en un mensaje de **petición SCVP**. Dicha solicitud puede requerir al servidor o bien la construcción de la ruta de certificación entre un certificado en cuestión y una raíz de confianza (DPD), o bien la validación del certificado y su correspondiente ruta de certificación asociada (DPV).

Por su parte, el servidor realiza los controles de validación de certificados estándar para asegurarse que el certificado en cuestión no haya expirado y que fuera emitido por una CA reconocida y de confianza. Para ello podría contar con unas **autoridades de certificación de confianza** almacenadas localmente en éste, que serían utilizadas para poder determinar la ruta de certificación si el cliente lo requiriese.

La respuesta por parte del servidor, que a su vez se encuentra encapsulada en un mensaje de **respuesta SCVP**, estará basada en el estado del certificado (válido o inválido) y/o la información adicional requerida por el solicitante.

La comunicación del dispositivo terminal en el que se encuentra el cliente con el servidor se realiza a través de conexiones HTTP (sección 2.3.1). A su vez, encapsulado dentro de estos mensajes de solicitud/respuesta HTTP se encuentra el tipo de mensaje que el protocolo SCVP establece para ello, como se detalla en la sección 2.2.4.

Resulta importante reseñar que el elemento fundamental sobre el que versa el sistema completo es el certificado digital. Este como ya se detalló en la sección 2.1.2, se trata de un documento electrónico firmado por una autoridad de certificación y que principalmente asocia la identidad de un sujeto (persona física, organismo o empresa) a una clave pública. Igualmente en esta sección, se encuentra indicado el formato de certificado utilizado para la realización de este proyecto, el certificado digital *X.509*.

3.2 Funcionalidad

El sistema, como ya se ha mencionado anteriormente, permite al usuario la realización de dos acciones sobre uno o varios certificados elegidos. Por un lado, permite delegar en el servidor la construcción de la ruta de certificación correspondiente al certificado escogido (DPD), y por otro lado ofrece a sus usuarios la opción de, además de construir la ruta de certificación, validar dicha ruta y por tanto el certificado en cuestión (DPV). Además el sistema cuenta con el envío de mensajes firmados digitalmente entre el cliente y el servidor.

A continuación se detallan los diferentes procesos criptográficos mencionados:

- **Construcción de la ruta de certificación**. El servidor permite al cliente calcular la ruta de certificación correspondiente de un certificado seleccionado por el cliente. La estructura de dicha ruta o cadena de certificación se puede observar en la **Figura 7**.

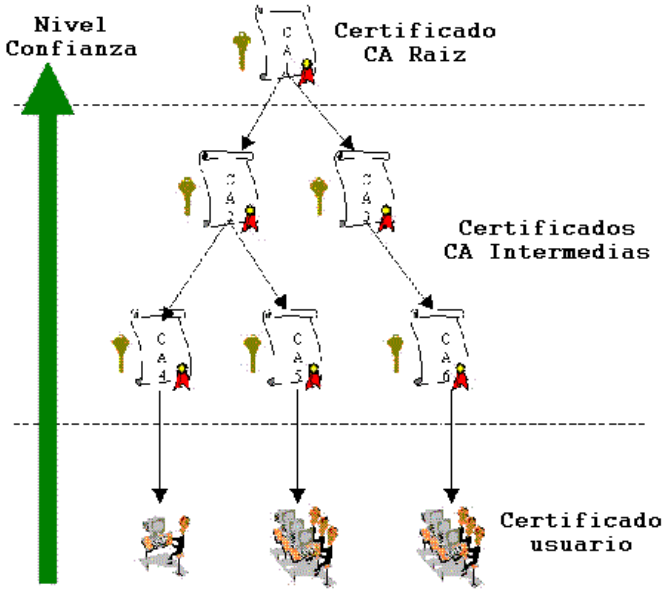

Figura 7. Cadena de certificación (Fuente [14])

- **Proceso de descubrimiento de la ruta de certificación.**
  De forma esquemática el algoritmo consiste en:

  i. Comenzar con el certificado de usuario a verificar, firmado con la clave pública de alguna entidad de certificación.

  ii. Si dicho certificado está firmado por una autoridad de certificación intermedia, debe continuar la búsqueda en vertical hacia arriba comprobando la firma del próximo certificado en la cadena de manera recursiva.

  iii. Si el certificado se encuentra firmado por una autoridad de certificación raíz o confiable, entonces el proceso de

búsqueda termina y el conjunto de todos los certificados previos conforman la ruta de certificación.

- **Validación de la ruta de certificación**. El servidor también permitirá al cliente delegar en él la validación de la ruta de certificación basándose en el procedimiento especificado en la RFC 5280 [2] . Es decir, verificar que la ruta sea consistente y que los certificados que la componen sean válidos.

    - **Proceso de validación de un certificado.**

        i. Comprobar la firma digital del certificado utilizando la clave pública contenida en el certificado emisor.

        ii. Comprobar que la fecha de referencia esté comprendida dentro del periodo de vigencia del certificado.

        iii. Comprobar que el certificado no se encuentre revocado o suspendido.

        iv. Comprobar que todos los certificados intermedios sean de entidades de certificación y no de usuarios finales.

        v. Comprobar que no exista ninguna extensión critica que no sea capaz de procesarse.

- **Firma digital.** Tanto los mensajes de solicitud enviados por el cliente como los de respuesta enviados por el servidor, pueden o deben según el caso, ir firmados digitalmente. Esto implica unas garantías de no repudio y de integridad del mismo, es decir, de conocimiento inequívoco de quién es el emisor del mensaje y de que el mensaje firmado es el original.

  El procedimiento tanto de firma digital de unos datos como de comprobación de dicha firma se encuentra esbozado en la siguiente **Figura 8**.

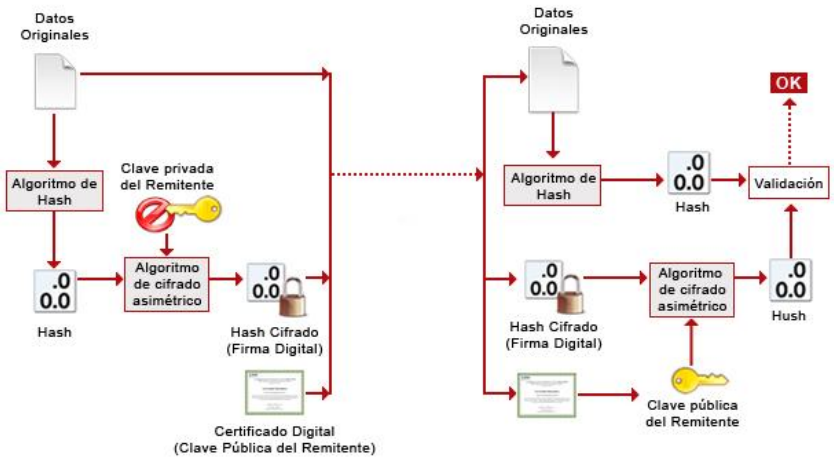

Figura 8. Proceso de creación/verificación de firma digital (Fuente [15])

- **Creación de un mensaje firmado.**
  Este proceso consiste en:

  i. Se calcula el valor de *hash* del mensaje (datos originales).

  ii. Se cifra el valor del *hash* con la clave privada del remitente.

  iii. Se añade al mensaje, el valor del *hash* cifrado como una firma digital.

  iv. Se añade al mensaje y al valor de la firma digital, el certificado digital del remitente.

- **Validación de un mensaje firmado.**
  Este proceso cuenta con los siguientes pasos, teniendo en cuenta que se recibe tanto el valor de la firma digital (*hash* cifrado), como el mensaje (datos originales) y el certificado digital del remitente.

  i. Se calcula el valor de *hash* del mensaje (datos originales).

  ii. Se obtiene la clave pública del remitente a través de su certificado digital.

  iii. Se obtiene con la clave pública del remitente el valor del *hash* cifrado.

  iv. Se compara el valor del *hash* calculado (i) y el valor del *hash* descifrado (iii).

  v. Si ambos valores coinciden el mensaje es válido.

3.3 Diseño de la aplicación

Con el fin de lograr el desarrollo simplificado del sistema y una escritura sencilla y reutilizable del código, se ha realizado una división de este en módulos.

Los módulos fundamentales de los que consta el sistema son la interfaz de usuario, el módulo correspondiente a la configuración del almacén de certificados, el módulo de política del servidor y el módulo criptográfico que comprende la construcción y validación de la ruta de certificación. Los dos primeros se encuentran implementados en el cliente, mientras que los otros dos se desarrollan en el servidor.

En la siguiente **Figura 9** se presentan dichos módulos y sus interacciones.

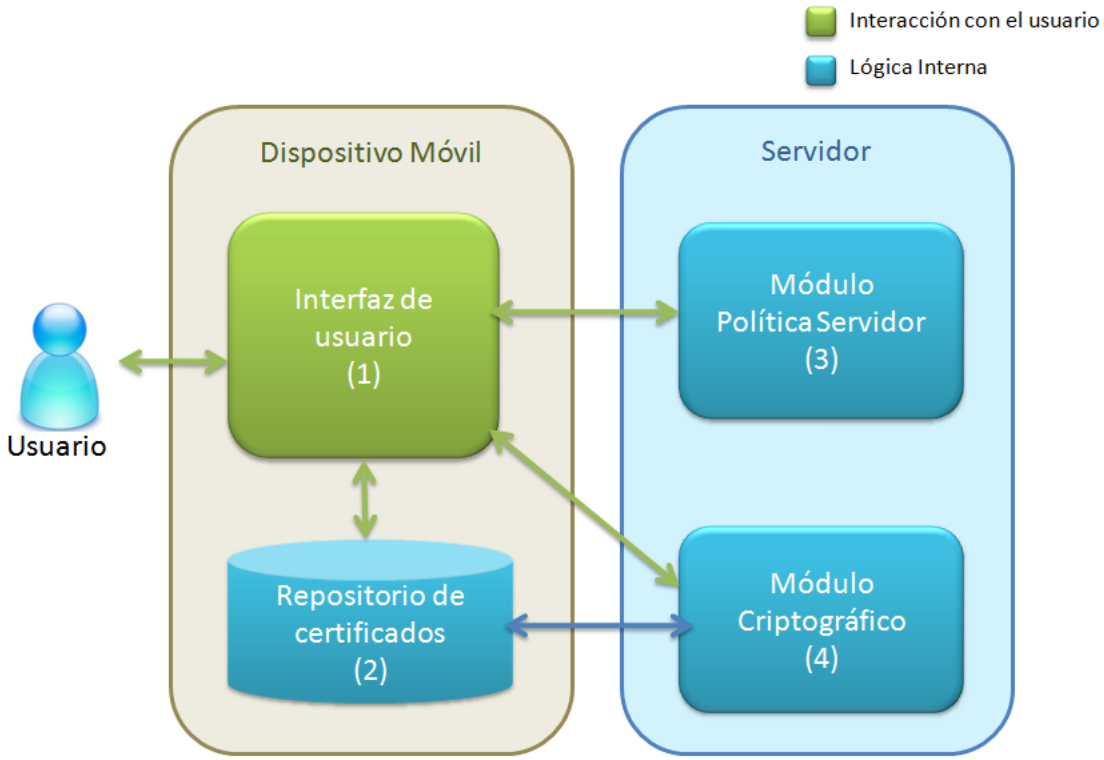

Figura 9. Diagrama de módulos de la aplicación

Si bien, cada uno de estos bloques será detallado en sucesivos capítulos, a continuación se realiza una breve descripción de cada uno de ellos que ayuda a una comprensión global del funcionamiento del sistema.

- **Módulo 1: Interfaz de usuario**

  La interfaz agrupa todas las vistas que la aplicación ofrece al usuario para su interacción con él.

- **Módulo 2: Repositorio de certificados de confianza del usuario**

  Es el módulo que permite al usuario la configuración del almacén de certificados de la aplicación con los que se realizarán las tareas del módulo 4. En él, se encuentran los certificados que el usuario desee añadir como comprobación en la ruta de certificación.

- **Módulo 3: Política del servidor**

  Es el módulo que permite al usuario consultar la política de validación que el servidor aplicará en las funciones del módulo criptográfico.

- **Módulo 4: Módulo Criptográfico**

  Es el módulo encargado tanto del cálculo de la ruta de certificación como de la validación de dicha ruta. Cuenta con un certificado o un conjunto de

certificados pertenecientes a una ruta como entradas al sistema para realizar la validación.

Tal y como se muestra en la **Figura 9**, dichos módulos se encuentran a su vez agrupados en dos capas: la capa de interacción con el usuario y la capa de lógica interna. De esta forma, se encontrarían separados la interfaz gráfica (correspondiente a la primera capa) de los módulos que dan la funcionalidad al sistema (englobados en la segunda capa).

3.4 Especificación de requisitos

3.4.1 Requisitos funcionales

Política del servidor	La aplicación permite al usuario consultar la política de validación del servidor
Certificado de usuario	La aplicación permite al usuario seleccionar un certificado digital sobre el que desea realizar las operaciones
Certificados de confianza	La aplicación permite al usuario seleccionar los certificados digitales de confianza para poder realizar las operaciones
Cálculo ruta de certificación	La aplicación permite al usuario calcular la ruta de certificación de un certificado digital previamente seleccionado
Validación	La aplicación permite al usuario calcular y validar la ruta de certificación de un certificado previamente seleccionado
Resultado ruta de certificación	La aplicación permite al usuario consultar el resultado satisfactorio o no satisfactorio del proceso de cálculo de la ruta de certificación sobre el certificado seleccionado. En el primer caso se permite al usuario acceder a dicha ruta.
Resultado validación	La aplicación permite al usuario consultar el resultado satisfactorio o no satisfactorio del proceso de validación sobre el certificado seleccionado
Envío de mensajes firmado	El sistema cuenta con la funcionalidad del envío de mensajes firmados digitalmente
Recepción de mensajes firmados	El sistema cuenta con la funcionalidad de procesado de mensajes firmados digitalmente
Certificados digitales	El sistema cuenta con la capacidad de envío, recepción y procesado de certificados digitales

Tabla 1. Requisitos funcionales

3.4.2 Requisitos no funcionales

Los requisitos no funcionales están a su vez divididos en: *requisitos de interfaz*, que indican como debe ser ésta; *requisitos de comunicación*, que fijan el modo de comunicación necesario dentro del sistema y *requisitos de operación*, que indican las restricciones y elementos necesarios para el correcto funcionamiento de este.

- **Requisitos de interfaz**

Interfaz intuitiva	La aplicación presenta una interfaz sencilla y accesible que permite al usuario seleccionar las diversas opciones de las que dispone el sistema
Aplicación en vertical	La orientación de la aplicación es vertical

Tabla 2. Requisitos no funcionales de interfaz

- **Requisitos de comunicación**

Protocolo HTTP	Las comunicaciones en el sistema se basan en la utilización del protocolo HTTP

Tabla 3. Requisitos no funcionales de comunicación

- **Requisitos de operación**

Instalación de certificados	El usuario debe instalar en el dispositivo móvil todos certificados digitales de los que desea hacer uso mediante la aplicación
Conexión	Es necesario disponer de una conexión para la transferencia de datos
Android	La aplicación garantiza su funcionamiento en dispositivos móviles con sistema operativo Android

Tabla 4. Requisitos no funcionales de operación

3.5 Casos de uso

Los casos de uso presentan el sistema y su funcionamiento a través de su utilización por parte del usuario de la aplicación. Este podrá realizar diversas acciones, las cuales conforman dos casos de uso. El caso 1, referente al módulo DPD encargado de la construcción de la ruta de validación y el caso 2, asociado al módulo DPV encargado de la validación de dicha ruta.

Los dos casos de uso, se describen en sus correspondientes tablas mostradas a continuación. En cada tabla se especifican los actores que intervienen en el caso, la descripción y el flujo que sigue el mismo.

- **Caso 1**. Construcción de la ruta de validación

Actores	Usuario
Descripción	Construcción de la ruta de certificación para su consulta y recepción.
Flujo	1. El usuario selecciona el certificado sobre el que desea calcular la ruta, los certificados de confianza propios y demás opciones sobre el tipo de resultado que desea 2. El usuario recibe y visualiza por pantalla la ruta de certificación en caso de que esta fuera construida por el servidor.

Tabla 5. Caso de uso 1: DPD

- **Caso 2**. Validación de un certificado digital

Actores	Usuario
Descripción	Construcción y validación de la ruta de certificación, para su consulta y recepción.
Flujo	1. El usuario selecciona el certificado sobre el que desea realizar la validación, los certificados de confianza propios y demás opciones sobre el tipo de resultado que desea 2. El usuario visualiza el resultado satisfactorio o no satisfactorio sobre la validación del certificado y la ruta de certificación en caso de que esta fuera construida por el servidor.

Tabla 6. Caso de uso 2: DPV

Capítulo 4

Implementación del sistema

4.1 Introducción

Una vez conocida, a través del capítulo anterior, la estructura del sistema a nivel global se procede a detallar en este capítulo cuatro su implementación. Dicha implementación ha sido dividida en dos partes, por un lado se encuentra la implementación de la interacción con el usuario y por otro lado, la implementación de la lógica interna.

Por lo tanto, en base al criterio previamente mencionado las clases y actividades que forman parte del sistema desempeñan funciones relacionadas puramente con la interfaz, con la lógica interna o bien con ambas las dos.

En el esquema de la siguiente **Figura 10** se encuentran representadas las clases y actividades desarrolladas para la construcción del sistema y las relaciones que estas mantienen entre ellas. No obstante, estos componentes y sus correspondientes métodos son explicados con más detalle en sucesivos apartados.

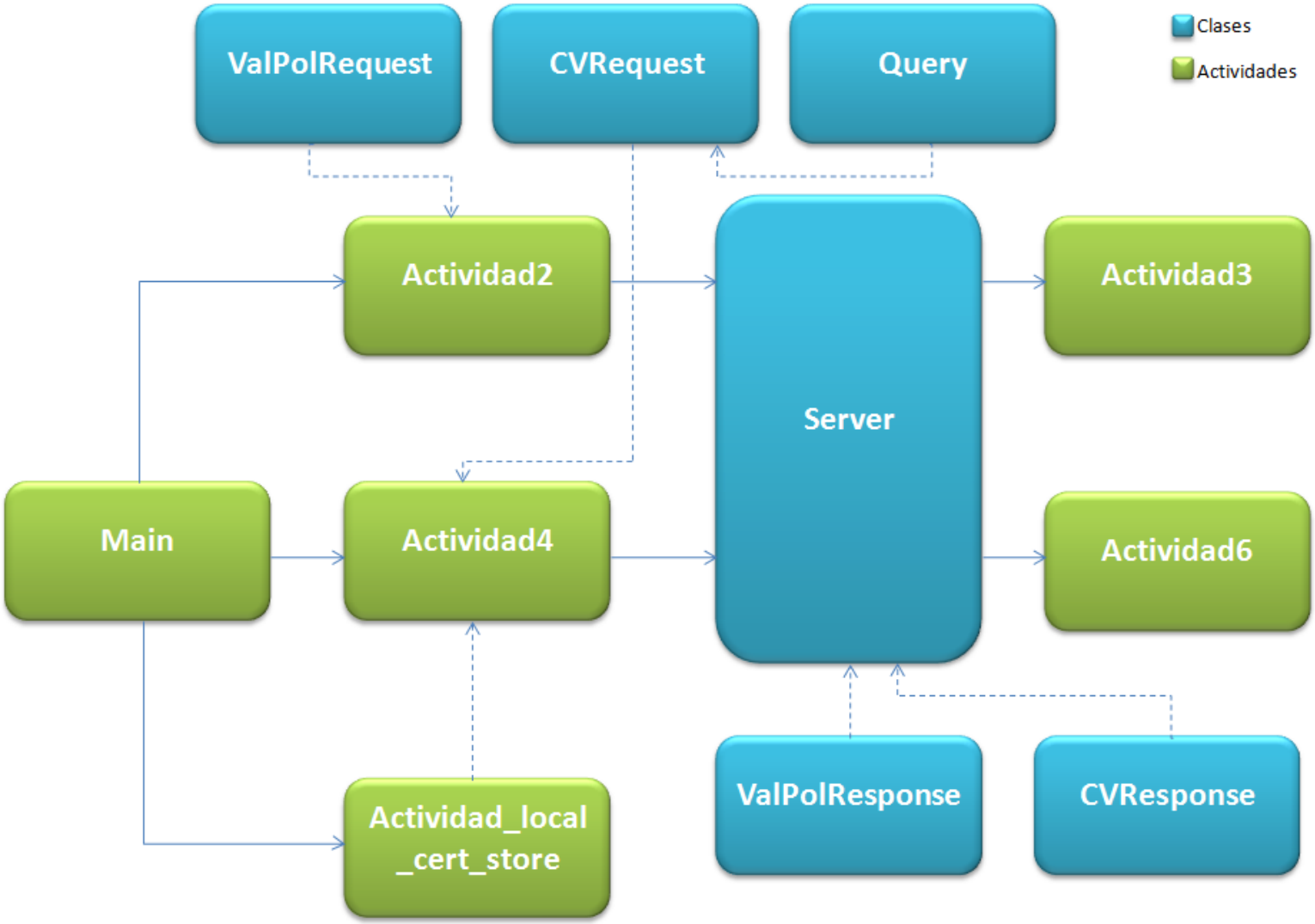

Figura 10. Clases y actividades del sistema

4.2 Interacción con el usuario

Para el correcto funcionamiento del sistema es imprescindible el diseño de una interfaz gráfica que permita la interacción del usuario con la aplicación.

La interfaz gráfica está formada por varias vistas, donde cada una de estas se corresponde con las distintas pantallas ofrecidas a lo largo del uso de la aplicación. La forma que el usuario posee de acceder a las funcionalidades que ofrece dicha aplicación, es a través de los elementos que conforman dichas vistas. Y las acciones que el usuario podrá realizar, son aquellas relacionadas con la consulta de la política al servidor, con la petición de construcción y/o validación de certificados por el servidor o con la gestión del almacén de certificados.

4.2.1 Interfaz de usuario

Como ya se especificó en secciones anteriores, la implementación de la parte correspondiente al cliente, se encuentra desarrollada en Android. Es decir, la interfaz gráfica del sistema está basada en el sistema operativo móvil Android.

En esta plataforma, una actividad es la componente principal encargada de mostrar al usuario la interfaz gráfica, es decir, sería el equivalente a una ventana, y es el medio de comunicación entre la aplicación y el usuario. En definitiva, una actividad es la clase Java que contiene el código necesario para llevar a cabo una determinada tarea.

Cada vista, la cual constituye el conjunto de elementos gráficos mostrados al usuario para su interacción con el sistema, se encuentra asociada a una actividad dentro de la cual se produce el despliegue de las pantallas y se da funcionalidad a los elementos que las conforman. Igualmente las actividades pueden contener código relacionado con la lógica del sistema.

Para la creación de la vista asociada, cada actividad hace uso del método *onCreate(Bunde savedInstanceState)*. Este método crea la actividad mediante la recepción de un parámetro *Bundle*, que contiene el estado anterior de la actividad para preservar la información que hubiera, en caso de que hubiera sido suspendida, o también puede iniciarse con *null* si bien dicha información no existiese.

Para el cambio entre pantallas, algunas actividades cuentan con el elemento *Intent*, encargado de pasar a la vista que corresponda según la acción tomada. Si bien las actividades son básicamente pantallas, los *Intent* son la manera de invocar a estas actividades. En definitiva, un *Intent* es, una clase que permite especificar una actividad a ejecutar. Sin embargo, cabe resaltar que aquellas actividades que se tratan de una vista final no contienen dicho elemento.

Tanto las actividades que intervienen en la creación de la interfaz del usuario como el flujo establecido entre ellas se encuentran recogidos en el esquema de la siguiente **Figura 11**.

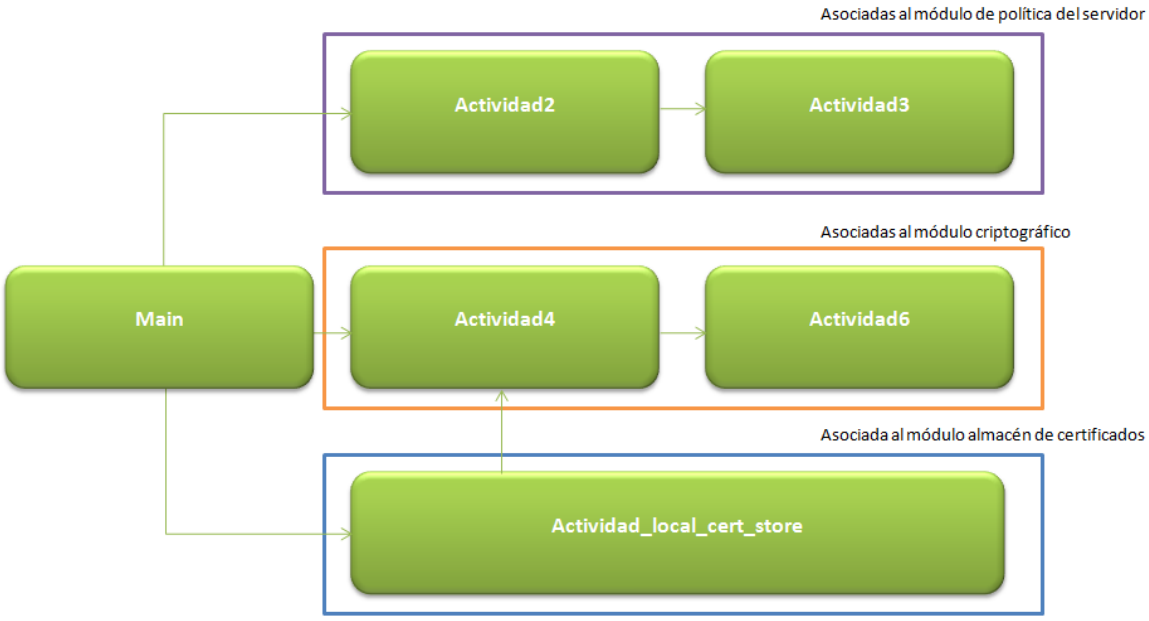

Figura 11. Actividades que componen la interfaz de usuario

4.2.1.1 Estructura de las vistas

Los recursos utilizados para la construcción de las diferentes vistas que conforman la interfaz de usuario son los explicados a continuación.

- **Layouts.** Ficheros creados en el directorio *res/layout* que contienen la definición de las vistas que conforman la interfaz. Estos ficheros permiten acciones tales como, situar los elementos gráficos dentro de la pantalla correspondiente del dispositivo o definir los parámetros visuales asociados a cada uno de ellos.

- **Values.** Ficheros creados en el directorio *res/values* utilizados para la definición de las constantes utilizadas como parámetros de los elementos de un *layout*. Existen tres tipos de ficheros definidos por defecto:

    - *dimens.xml:* contiene las dimensiones definidas por defecto a los elementos de una vista.

    - *strings.xml*: contiene cadenas de caracteres fijas.

    - *styles.xml*: contiene los estilos definidos por defecto a los elementos de una vista.

- **Drawable.** Ficheros creados en el directorio *res/drawable* que contienen los recursos gráficos utilizados como imágenes dentro de las pantallas. Dichos recursos pueden ser almacenados en diferentes subcarpetas en función de la resolución y densidad de la pantalla. Estas subcarpetas están organizadas desde densidad baja a densidad muy alta de la siguiente manera:

- *drawable-hdpi*
- *drawable-ldpi*
- *drawable-mdpi*
- *drawable-xhdpi*
- *drawable-xxhdpi*

- **AndroidManifest.xml.** Archivo ubicado en la raíz del proyecto, generado automáticamente y modificable. Es un archivo de configuración donde se pueden aplicar las configuraciones básicas de la aplicación. En *AndroidManifest* se declaran todas las especificaciones de la aplicación tales como las actividades, los *Intents*, bibliotecas, nombre de la aplicación etc.

En concreto para el desarrollo de la interfaz gráfica de la aplicación en cuestión se hace uso de algunos de dichos ficheros.

Primeramente, la aplicación se encuentra formada por seis *layouts*, es decir, cuenta con seis vistas distintas. Es el fichero *main.xml*, el fichero de partida de la aplicación (la vista principal) a partir del cual se accede al resto de pantallas, tal como se muestra en la siguiente **Figura 12**.

La aplicación también cuenta con los ficheros, mencionados anteriormente para la definición de constantes: *strings.xml*, *styles.xml* y *dimens.xml*.

Finalmente, el proyecto cuenta con varios recursos de tipo *drawable*, consistentes en imágenes con extensión *.png* o *.jpeg* , así como archivos *.xml* que definen el formato de algunos elementos de tipo *Button* de la aplicación.

4.2.1.2 Descripción de las vistas y acciones asociadas

A continuación, se proporciona una descripción detallada de cada una de las seis vistas incluidas en el programa, mediante la descripción de los *layouts*. Esta descripción acompañada por el esquema de la **Figura 12**, pretende ilustrar las opciones de configuración disponibles a través de la interfaz diseñada y como tiene lugar la navegación a través de las pantallas. Además todas las vistas expuestas a continuación se encuentran recogidas en el **Anexo C.**

- **Vista del layout *main.*** Esta vista contiene el menú de inicio de la aplicación. A través de él, se accede a la parte del almacén de certificados, a la parte de consulta de política del servidor o a la parte de construcción/validación de certificados. La vista se compone de los siguientes elementos:
  - Botón *Repositorio de certificados*: proporciona acceso al layout *activity_actividad_local_cert_store.*
  - Botón *Política del servidor*: proporciona acceso al layout a*ctivity_actividad2.*
  - Botón *Validación:* proporciona acceso al layout *activity_actividad4.*

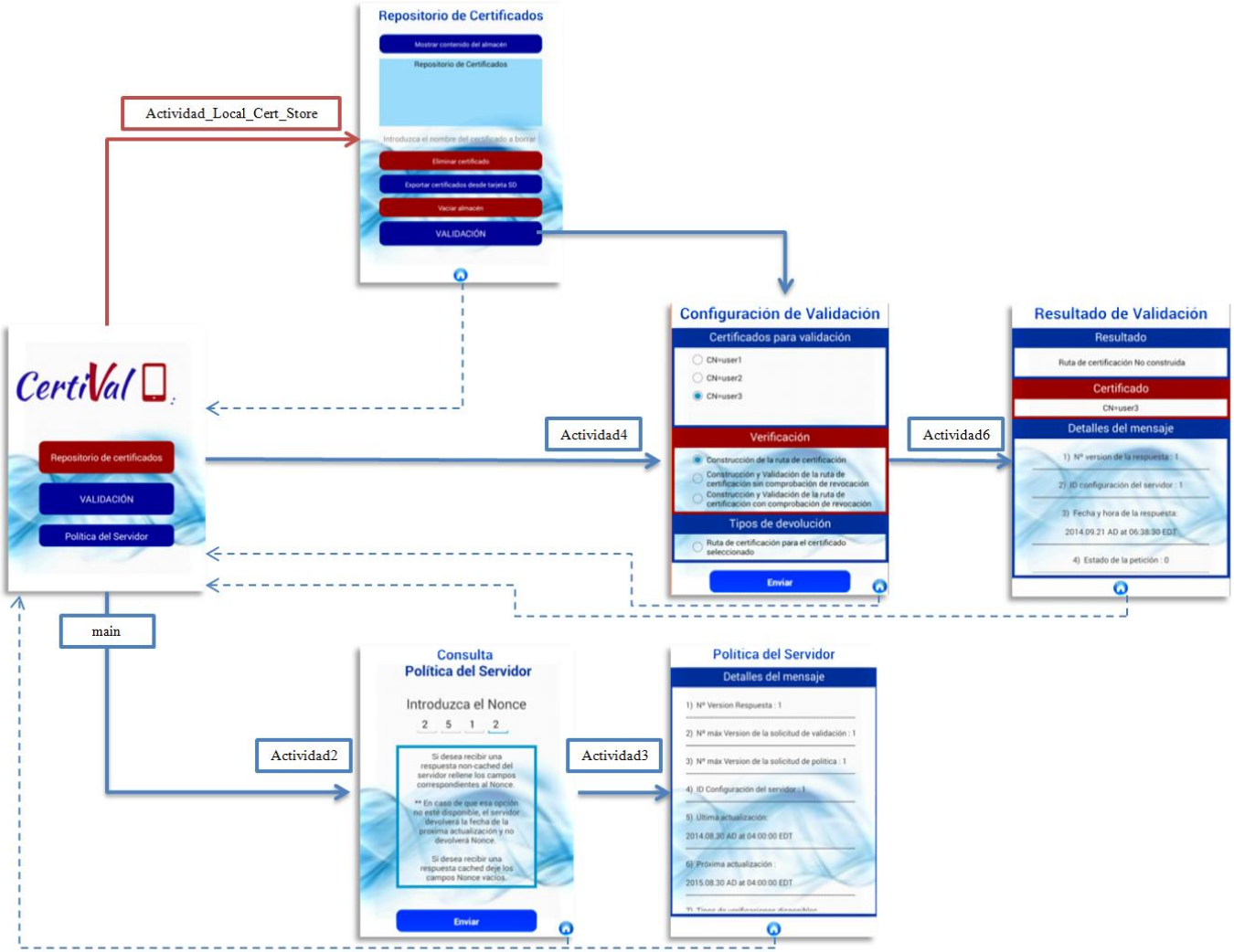

Figura 12. Vistas de la aplicación y sus relaciones

- **Vista del layout *activity_actividad2.*** Esta vista contiene los elementos para la consulta de la política de validación del servidor. Los elementos por los que está formada son los siguientes:

    - Receptores de texto *nonce:* recogen el número nonce introducido por el usuario.

    - Visor de texto *rnexplanation*: muestra al usuario las instrucciones a seguir para la consulta de la política.

    - Botón *Enviar:* proporciona acceso al layout *activity_actividad3.*

    - Icono *Inicio:* proporciona acceso al layout *main.*

- **Vista del layout *activity_actividad3.*** Esta vista contiene el resumen correspondiente a la política enviada por el servidor. La vista se compone de los siguientes elementos:

    - Visor de texto *valPolResponse:* recoge la política enviada por el servidor para ser mostrada al usuario.

    - Icono Inicio: proporciona acceso al layout *main.*

- **Vista del layout *activity_actividad_local_cert_store.*** En esta vista se permite la configuración del almacén de certificados al usuario y para ello se despliegan los siguiente elementos:

    - Botón *Mostrar contenido del almacén:* recorre la lista de certificados disponibles en el almacén y muestra dicha información en el visor de texto *list*.

    - Visor de texto *list:* recoge la lista de los certificados almacenados en el repositorio.

    - Editor de texto *delete*: recoge el nombre del certificado a borrar introducido por el usuario.

    - Botón *Eliminar certificado*: elimina el certificado a borrar introducido por el usuario.

    - Botón *Exportar certificados desde tarjeta SD*: copia los certificados almacenados en la tarjeta SD en el repositorio interno de la aplicación.

    - Botón *Vaciar almacén*: elimina los certificados almacenado en el repositorio interno de la aplicación.

    - Botón *Validación*: proporciona acceso al layout *activity_actividad4.*

    - Icono *Inicio*: proporciona acceso al layout *main.*

- **Vista del layout *activity_actividad4.*** En esta vista se permite la configuración del tipo de petición de construcción de la ruta y/o validación de un certificado que se va a realizar al servidor y para ello se despliegan los siguientes elementos:

- Visor de texto *queriedCerts*: recoge dinámicamente los certificados almacenados en el repositorio externo que el usuario puede seleccionar para ser validados.

- Botones Radio *radioChecks*: cada uno de ellos recogen las tres opciones que el usuario puede aplicar sobre el certificado: construcción de la ruta, validación de la ruta sin revocación y validación de la ruta con revocación.

- Visor de texto *wantBackOptions*: recoge una opción u otra según el botón *radioChecks* que haya sido seleccionado.

- Botón *Enviar:* proporciona acceso al layout *activity_actividad6*

- Icono *Inicio*: proporciona acceso al layout *main.*

- **Vista del layout *activity_actividad6.*** Esta vista contiene el resumen correspondiente al resultado de la construcción de la ruta de certificación y/o validación enviada por el servidor. La vista se compone de los siguientes elementos:

  - Visor de texto *Resultado de validación:* recoge un mensaje resumen sobre el resultado satisfactorio o insatisfactorio de la construcción y/o validación del certificado.

  - Visor de texto *userCertName*: recoge el *SubjectDN* del certificado sobre el que se realizaron las acciones en el servidor.

  - Visor de texto *cvResponseText:* recoge los detalles del mensaje enviado por el servidor sobre la construcción y/o validación del certificado.

  - Icono *Inicio*: proporciona acceso al layout *main.*

4.3 Lógica interna del sistema

Una vez conocemos cómo interactúa el usuario con la aplicación, se precisa conocer las acciones que se ejecutan internamente para que el programa funcione. A continuación se describe la lógica interna del sistema, que se encuentra divida en tres módulos: el módulo de política del servidor, el módulo criptográfico y el módulo correspondiente al repositorio de certificados.

Para cada módulo se explican las clases utilizadas, su funcionamiento y la comunicación entre ellas.

4.3.1 Implementación del módulo política del servidor

El módulo de política del servidor es el encargado de la generación y procesado de la política de validación del servidor. Es decir, este módulo trata el par de mensajes solicitud/respuesta establecido por el protocolo SCVP para la consulta de la política de validación desde el cliente al servidor.

Dicho módulo, comienza en el mensaje de petición de política enviado desde el cliente, pasa por el procesado de dicha petición y generación de la respuesta y finaliza en la muestra de dicha política al usuario por medio de la interfaz.

Por un lado, las clases *ValPolRequest* y *ValPolResponse* permiten definir la estructura del mensaje de solicitud y de respuesta asociado a la política respectivamente. Además la clase *ServerJ.java* ejecuta la lógica implementada por el servidor.

Por otro lado, las actividades empleadas en este modulo para el desarrollo de las tareas asociadas son: *Actividad2* y *Actividad3*. Dichas actividades pertenecen a la lógica del cliente.

Las clases y actividades previamente mencionadas con las que cuenta este módulo, así como las relaciones que estas mantienen entre ellas se ponen de manifiesto en el esquema de la siguiente **Figura 13**.

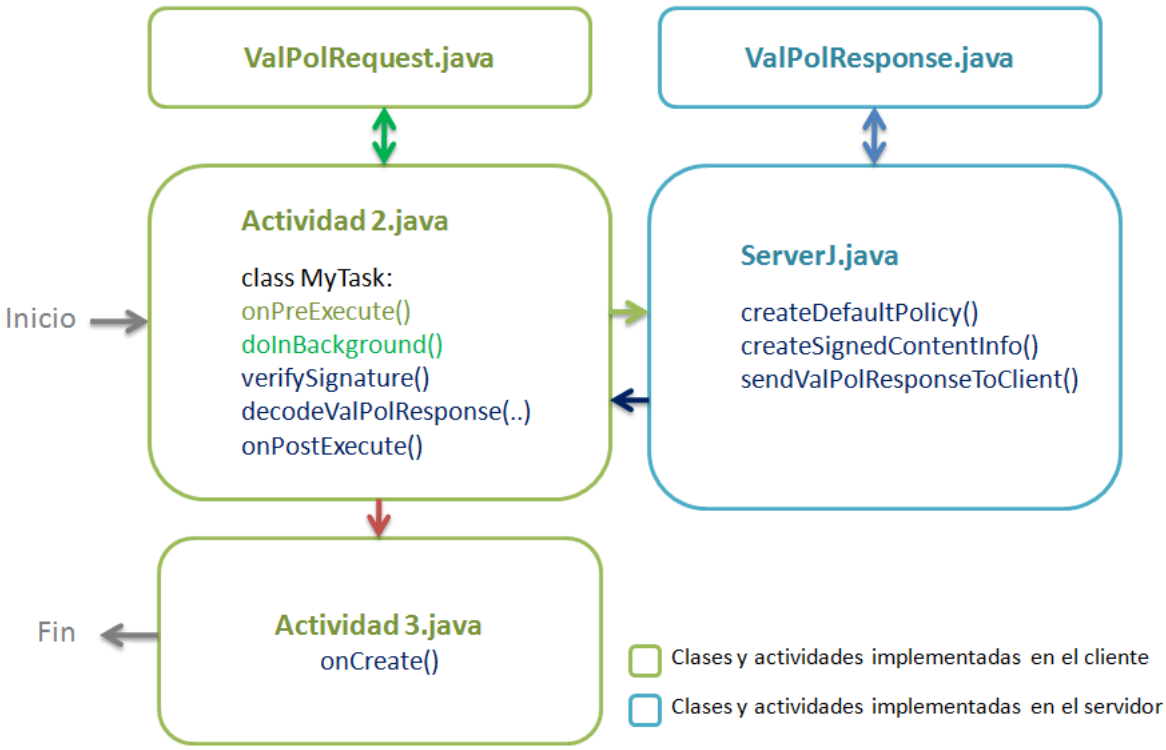

Figura 13. Clases y actividades del módulo política del servidor

4.3.1.1 Estructura del mensaje de solicitud

En primer lugar se determinan los datos que componen el mensaje de solicitud que el cliente envía al servidor a fin de conocer la política de validación que éste aplica.

El nombre que recibe la estructura de datos especificada para dicho mensaje de solicitud es el de *ValPolRequest*, según está definido en la RFC 5055 [5], siendo los campos que componen dicha estructura de datos los que se especifican en la siguiente **Tabla 7**. Cabe mencionar que el formato de definición para la estructura de datos es la notación ASN.1 que se encuentra detallada en la sección 2.2.2.

Campo	Tipo de dato	Tipo de campo	Descripción
ValPolRequest	Sequence	-	Estructura fundamental
vpRequestVersion	Integer	**Oblig.**	Versión de la petición
requestNonce	Octet String	**Oblig.**	Identificador de solicitud del usuario

Tabla 7. Campos de la estructura del mensaje solicitud de política

El cliente, en este caso la aplicación debe rellenar los dos campos correspondientes a la estructura *ValPolRequest* para el envío correcto de la petición de política al servidor. En el sistema implementado, la creación de dicha estructura se realiza en la *Actividad2* siendo el valor del primer campo uno y el valor del segundo campo el valor correspondiente introducido por el usuario por pantalla.

4.3.1.2 Estructura del mensaje de respuesta

En segundo lugar se determinan los datos que componen el mensaje de respuesta que el servidor envía al cliente y que contiene la política de validación que este aplica.

El nombre que recibe la estructura de datos especificada para dicho mensaje de respuesta es el de *ValPolResponse*, según está definido en la RFC 5055 [5], siendo los campos que componen dicha estructura de datos los que se especifican en la siguiente **Tabla 8**. Cabe mencionar que al igual que en la sección anterior, el formato de definición para la estructura de datos es la notación ASN.1 que se encuentra detallada en la sección 2.2.2.

El servidor debe rellenar los campos correspondientes a la estructura *ValPolResponse* para el envío correcto de la política al cliente.

En el sistema implementado, la creación de dicha estructura se realiza en la clase *ServerJ.java* por medio del método *createDefaultPolicy().* En dicho método, el sistema rellena todos los campos a excepción del campo *requestNonce,* ya que la funcionalidad de enviar respuestas almacenadas en caché en este sistema, está desactivada. Cabe resaltar que dicho campo está especificado como opcional en el protocolo SCVP. El método en concreto devuelve el mismo tipo de estructura *ValPolResponse* para cada petición del usuario.

Campo	Tipo de dato	Tipo de campo	Descripción
ValPolResponse	Sequence	**Oblig.**	Estructura fundamental
vpResponseVersion	Integer	**Oblig.**	Nº de versión de la respuesta
maxCVRequestVersion	Integer	**Oblig.**	Nº máximo de versión de la petición de validación
maxVPRequestVersion	Integer	**Oblig.**	Nº máximo de versión de la petición de política
serverConfigurationID	Integer	**Oblig.**	ID de configuración del servidor
thisUpdate	GeneralizedTime	**Oblig.**	Última actualización
nextUpdate	GeneralizedTime	**Oblig.**	Próxima actualización
supportedChecks	CertChecks	**Oblig.**	Tipos de verificaciones disponibles
supportedWantBacks	WantBack	**Oblig.**	Tipos de devoluciones disponibles
validationPolicies	Sequence	**Oblig.**	Política de validación
validationAlgs	Sequence	**Oblig.**	Algoritmos de validación
authPolicies	Sequence	**Oblig.**	Políticas de autenticación
responseTypes	ResponseTypes	**Oblig.**	Tipos de respuesta disponibles
defaultPolicyValues	RespValidationPolicy	**Oblig.**	Valores de política por defecto
revocationInfoTypes	RevocationInfoTypes	**Oblig.**	Tipos de comprobaciones de revocación
signatureGeneration	Sequence	**Oblig.**	Algoritmos de firma digital utilizados por el servidor
signatureVerification	Sequence	**Oblig.**	Algoritmos de firma digital que el servidor puede verificar
hashAlgorithms	Sequence	**Oblig.**	Algoritmos de Hash que el servidor utiliza
serverPublicKeys	Sequence	Opci.	Claves públicas del servidor
clockSkew	Integer	**Oblig.**	Máximo número de minutos de sesgo del reloj
requestNonce	Octet String	Opci.	Identificador del usuario

Tabla 8. Campos de la estructura del mensaje respuesta de política

4.3.1.3 Comunicación y procesos

Por último, una vez establecidas las estructuras de mensaje necesarias para llevar a cabo la ejecución de dicho módulo, se procede a explicar el proceso de comunicación que se lleva a cabo entre el cliente y el servidor para dicho fin.

La comunicación se establece para todos los pares de solicitud/respuesta del protocolo, no sólo para el referido a la política, sobre el protocolo HTTP detallado en la sección 2.3.1. Es decir, el intercambio de mensajes que se realiza entre el cliente y el servidor se establece en base a dicho protocolo, de tal manera que las estructuras previamente especificadas y encapsuladas como se detalla en la secciones 2.2.4.4 y 2.2.4.5, se incluyen dentro del contenido de un mensaje de petición o respuesta de tipo HTTP según proceda.

A continuación se listan de manera ordenada los métodos que deben ejecutarse, y por lo tanto los procesos que se deben llevar a cabo, tanto en el lado del cliente como en el lado del servidor para el correcto funcionamiento de dicho módulo. El funcionamiento especificado a continuación es realizado en base al diagrama de la **Figura 13**. Es importante recordar que del lado del cliente se ejecutan las *Actividad2* y *Actividad3*, y del lado del servidor se ejecuta la clase *ServerJ*.

1. *Actividad2* lanza el método *doInBackGround()* de la clase *MyTask* donde se crea *ValPolRequest* y se encapsula.

2. La clase *ServerJ* se mantiene a la escucha de peticiones. Recibe la petición de política e inmediatamente ejecuta los siguientes procesos:

    i. *ServerJ* lanza el método *sendValPolResponseToClient(...)* donde se decodifica el mensaje de solicitud

    ii. *sendValPolResponseToClient(...)* lanza el método *createDefaultPolicy()* para crear el mensaje de respuesta con la política

    iii. *sendValPolResponseToClient(...)* lanza el método *createSignedContentInfo(...)* para firmar digitalmente el mensaje.

    iv. *sendValPolResponseToClient(...)* envía dicho mensaje encapsulado.

3. *Actividad2* recibe el mensaje de respuesta e inmediatamente ejecuta los siguientes procesos:

    i. Por medio de la clase *MyTask*, el método *doInBackground(...)* lanza el método *verifySignature(...)* para verificar la firma digital del mensaje

    ii. *doInBackground(...)* lanza el método *decodeValPolResponse(...)* que decodifica el mensaje de respuesta y lo almacena en un *String*

    iii. Se ejecuta el método *onPostExecute(...)* donde se envía dicho *String* a la *Actividad3*.

4. *Actividad3* lanza el método *onCreate(...)* donde se recupera dicho *String* con la política y se muestra al usuario por pantalla.

4.3.2 Implementación del módulo criptográfico

El módulo criptográfico es el encargado de realizar los procedimientos tanto de construcción de rutas como de validación de certificados. Por lo tanto, este módulo trata el par de mensajes solicitud/respuesta establecido por el protocolo SCVP para dichas tareas.

Dicho módulo, comienza con el mensaje de petición de validación enviado desde el cliente, pasa por el procesado de dicha petición, realización de las tareas, generación de la respuesta y finaliza con la muestra del resultado de dicha acción usuario por medio de la interfaz.

Por un lado, las clases implementadas *CVRequest*, *Query* y *CVResponse* permiten definir la estructura del mensaje de solicitud y de respuesta asociado a la validación respectivamente. Además la clase *ServerJ.java* es la encargada de toda la lógica implementada por el servidor.

Por otro lado, las actividades empleadas en este módulo para el desarrollo de las tareas asociadas son: *Actividad4* y *Actividad6*. Dichas actividades pertenecen a la lógica del cliente.

Las clases y actividades previamente mencionadas con las que cuenta este módulo, así como las relaciones que estas mantienen entre ellas se evidencia en el esquema de la siguiente **Figura 14**.

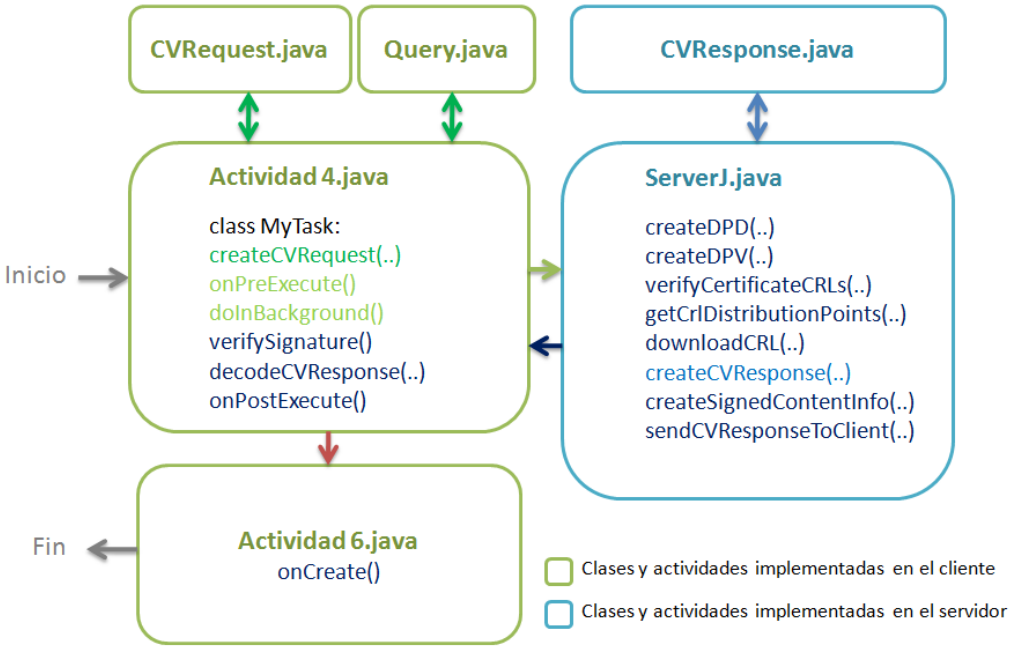

Figura 14. Clases y actividades del módulo criptográfico

4.3.2.1 Estructura del mensaje de solicitud

En primer lugar se determinan los datos que componen el mensaje de solicitud que el cliente envía al servidor con el fin de realizar o bien la construcción de una ruta o bien la construcción y validación de una ruta asociada a un certificado digital.

El nombre que recibe la estructura de datos especificada para dicho mensaje de solicitud es el de *CVRequest*, según está definido en la RFC 5055 [5], siendo los campos que componen dicha estructura los que se especifican en la siguiente **Tabla 9**. Cabe mencionar que el formato de definición para la estructura de datos es la notación ASN.1 que se encuentra detallada en la sección 2.2.2.

Campo	Tipo de dato	Tipo de campo	Descripción
CVRequest	Sequence	-	Estructura fundamental
cvRequestVersion	Integer	**Oblig.**	Número de versión de la solicitud
query	Query (Detallado a continuación)	**Oblig.**	Campo que adjunta el certificado en cuestión entre otros elementos
requestorRef	GeneralNames	Opci.	Lista de nombre que identifican los servidores SCVP
requestNonce	Octet String	Opci.	Numero de identificador de petición generado por el cliente. Indica preferencia para recibir respuestas no almacenadas en caché
requestorName	GeneralName	Opci.	Identificador del solicitante
responderName	GeneralName	Opci.	Identificador del servidor que el cliente desea que firme la respuesta
requestExtensions	Extensions	Opci.	Extensiones
signatureAlg	AlgorithmIdentifier	Opci.	Algoritmo con el que el servidor debería firmar la respuesta
hashAlg	Object Identifier	Opci.	Algoritmo de hash con el que el servidor debería general el hash de la respuesta
requestorText	UTF8String	Opci.	Petición de texto en la respuesta

Tabla 9. Campos de la estructura del mensaje de solicitud de validación

El cliente, en este caso la aplicación debe rellenar los campos correspondientes a la estructura *CVRequest* para el envío correcto de la petición de validación al servidor. En el sistema implementado, la creación de dicha estructura se realiza en la *Actividad4*, por medio de la ejecución del método *createCVRequest(...)*.

En dicho método se rellenan, gracias a la información seleccionada por el usuario por pantalla, los dos campos obligatorios para esta sección. Es decir, en el método *createCVRequest()* se genera un tipo de estructura de dato *CVRequest* cuyos campos son *cvRequestVersion* y *query*. Dada la complejidad estructural del segundo tipo de dato, su estructura se encuentra detallada en la **Tabla 10**.

Para la creación de la estructura de datos *Query* que a su vez se encuentra incluida en la estructura de datos fundamental para el mensaje de petición de validación *CVRequest*, se han incorporado los siguientes campos: *queriedCerts*, *checks*, *wantback*, *validationPolicy*, *intermediateCerts.*

Estos campos son cumplimentados gracias a la información que se recibe del usuario por pantalla. En este caso, tal como se puede observar en los elementos listados en el párrafo anterior, dicha estructura *Query* incluye tanto campos definidos como obligatorios y algunos opcionales.

Se le ofrece al usuario poder seleccionar un tipo de devolución de resultado que desee (*wantBack*) y adjuntar los certificados intermedios de confianza para que el servidor realice correctamente la construcción de la ruta de certificación (*intermediateCerts*). Esta ruta se construye también gracias a los certificados incluidos en el campo *trustAnchors* de *validationPolicy* que se puede consultar en la sección 3.2.4 de la RFC 5055 [5].

Campo	Tipo de dato	Tipo de campo	Descripción
Query	Sequence	-	
queriedCerts	CertReferences	**Oblig.**	Certificado/s sujeto de la petición/respuesta
checks	CertChecks	**Oblig.**	Comprobaciones que el usuario desea realizar sobre el certificado
wantBack	WantBack	Opci.	Tipos de devoluciones que el usuario desea obtener del resultado
validationPolicy	ValidationPolicy	**Oblig.**	Política de validación
responseFlags	ResponseFlags	Opci.	Características opcionales que el servidor debe incluir en la respuesta
serverContextInfo	Octet String	Opci.	Contexto sobre una solicitud/respuesta previa
validationTime	GeneralizedTime	Opci.	Fecha y hora a la que el cliente desea que el servidor realice las operaciones
intermediateCerts	CertBundle	Opci.	Certificados intermedios de confianza
revInfos	RevocationInfos	Opci.	Tipos de revocación requeridos
producedAt	GeneralizedTime	Opci.	Fecha y hora más temprana para la respuesta en caso de que esta esté almacenada en caché.
queryExtensions	Extensions	Opci.	Extensiones

Tabla 10. Campos de la estructura de datos Query

4.3.2.2 Estructura del mensaje de respuesta

En segundo lugar se determinan los datos que componen el mensaje de respuesta que el servidor envía al cliente y que contiene el resultado asociado a la acción requerida por el cliente. Es decir, dicho mensaje encapsula el resultado satisfactorio o insatisfactorio al proceso de construcción de ruta o validación realizado en el servidor.

El nombre que recibe la estructura de datos especificada para dicho mensaje de respuesta es el de *CVResponse*, según está definido en la RFC 5055 [5], siendo los campos que componen dicha estructura los que se especifican en la siguiente **Tabla 11**. Cabe mencionar que al igual que en la sección anterior, el formato de definición para dicha estructura de datos es la notación ASN.1 que se encuentra detallada en la sección 2.2.2.

Campo	Tipo de dato	Tipo de campo	Descripción
CVResponse	Sequence	-	Estructura fundamental
cvResponseVersion	Integer	**Oblig.**	Nº de versión de la respuesta
serverConfigurationID	Integer	**Oblig.**	Id de versión del servidor que proceso la petición
producedAt	GeneralizedTime	**Oblig.**	Fecha y hora de generación de la respuesta
responseStatus	ResponseStatus	**Oblig.**	Información de estado de la respuesta
respValidationPolicy	RespValidationPolicy	Opci.	Referencia a la política de validación utilizada
requestRef	RequestReference	Opci.	Identificador de la respuesta con la petición
requestorRef	GeneralNames	Opci.	Identificador del solicitante original. Aplicable en SCVP Relay.
requestorName	GeneralNames	Opci.	Identidad asociada a un cliente con varias identidades
replyObjects	ReplyObjects	Opci.	Devuelve los objetos requerido por el cliente en la petición
respNonce	Octet String	Opci.	Identificador de solicitud/respuesta
serverContextInfo	Octet String	Opci.	Información de contexto
cvResponseExtensions	Extensions	Opci.	Extensiones
requestorText	UTF8String	Opci.	Texto requerido por el cliente

Tabla 11. Campos de la estructura del mensaje de respuesta de validación

El servidor debe rellenar los campos correspondientes a la estructura *CVResponse* arriba mostrada para el correcto envío del correspondiente mensaje de respuesta de validación.

En el sistema implementado, la creación de dicha estructura se realiza en la clase *ServerJ.java* por medio del método *createCVResponse(...).* En dicho método el servidor rellena los campos obligatorios y un campo opcional referente a la estructura *CVResponse* en relación a la operación realizada.

En definitiva, son cumplimentados los datos: *cvResponseVersion*, *serverConfigurationID*, *producedAt* y *responseStatus* como campos obligatorios y *replyObjects* como campo opcional. El sistema cumplimenta este último campo con el fin de proporcionar una respuesta más amplia y completa en relación a las operaciones de construcción y/o validación realizadas.

4.3.2.3 Comunicación y procesos

Por último, una vez establecidas las estructuras de mensaje necesarias para llevar a cabo la ejecución de dicho módulo, se procede a explicar el proceso de comunicación que se realiza entre el cliente y el servidor para dicho fin.

La comunicación se establece para todos los pares de solicitud/respuesta del protocolo sobre el protocolo HTTP explicado en la sección 2.3.1. Es decir, el intercambio de mensajes que se realiza entre el cliente y el servidor se establece en base a dicho protocolo, de tal manera que las estructuras previamente especificadas y encapsuladas como se detalla en la secciones 2.2.4.6 y 2.2.4.7, se incluyen dentro del contenido de un mensaje de petición o respuesta de tipo HTTP según proceda.

A continuación se listan de manera ordenada los métodos que deben ejecutarse, y por lo tanto los procesos que se deben llevar a cabo, tanto en el lado del cliente como en el lado del servidor para el correcto funcionamiento de dicho módulo. El funcionamiento especificado a continuación es realizado en base al diagrama de la **Figura 14**. Es importante recordar que del lado del cliente se ejecutan las *Actividad4* y *Actividad6*, y del lado del servidor se ejecuta la clase *ServerJ*.

Dado que el módulo criptográfico cuenta con la implementación de tres tipos de acciones diferentes, los procesos a seguir para la generación de mensajes tanto en el cliente como en el servidor serán distintos y por lo tanto, también lo serán los métodos a ejecutarse. Es decir, la lógica de procedimiento para el envío de solicitud y recepción de la correspondiente respuesta serán distintos si se trata de la construcción de la ruta de validación o DPD o de la validación de certificados digitales con o sin comprobación de revocación.

Listado ordenado de procesos para la construcción de ruta:

1. *Actividad4* lanza el método *doInBackGround()* de la clase *MyTask* desde donde se lanza el método create*CVRequest*. Este genera la estructura *CVRequest* y la encapsula.

2. La clase *ServerJ* se mantiene a la escucha de peticiones. Recibe la petición de construcción de ruta e inmediatamente ejecuta los siguientes procesos:

i. *ServerJ* lanza el método *sendCVResponseToClient(...)* donde se decodifica el mensaje de solicitud

ii. *sendCVResponseToClient(...)* lanza el método *createDPD()* para la construcción de la ruta de validación

iii. *sendCVResponseToClient(...)* lanza el método *createCVResponse()* para crear el mensaje de respuesta.

iv. *sendValPolResponseToClient(...)* lanza el método *createSignedContentInfo(...)* para firmar digitalmente el mensaje.

3. *sendCVResponseToClient(...)* envía dicho mensaje encapsulado.

4. *Actividad4* recibe el mensaje de respuesta e inmediatamente ejecuta los siguientes procesos:

   i. Por medio de la clase *MyTask*, el método *doInBackground(...)* lanza el método *verifySignature(...)* para verificar la firma digital del mensaje

   ii. *doInBackground(...)* lanza el método *decodeCVResponse(...)* que devuelve en una lista *String* tanto el mensaje de respuesta, como un mensaje informativo del resultado y el nombre del certificado sobre el que se realizó la acción.

   iii. Se ejecuta el método *onPostExecute(...)* donde se envía dicha lista a la *Actividad6*.

5. *Actividad6* lanza el método *onCreate(...)* donde se recuperan los tres mensajes de tipo *String* y se muestran al usuario por pantalla.

Listado ordenado de procesos para la validación sin revocación:

1. *Actividad4* lanza el método *doInBackGround()* de la clase *MyTask* desde donde se lanza el método create*CVRequest*. Este genera la estructura *CVRequest* y la encapsula para su envío.

2. La clase *ServerJ* se mantiene a la escucha por el socket. Recibe la petición de validación e inmediatamente ejecuta los siguientes procesos:

   i. *ServerJ* lanza el método *sendCVResponseToClient(...)* donde se decodifica el mensaje de solicitud

   ii. *sendCVResponseToClient(...)* lanza el método *createDPV()* para la construcción de la ruta y validación de esta.

   iii. *sendCVResponseToClient(...)* lanza el método *createCVResponse()* para crear el mensaje de respuesta.

   iv. *sendValPolResponseToClient(...)* lanza el método *createSignedContentInfo(...)* para firmar digitalmente el mensaje.

v. *sendCVResponseToClient(...)* envía dicho mensaje encapsulado por el socket.

3. *Actividad4* recibe el mensaje de respuesta por el socket e inmediatamente ejecuta los siguientes procesos:

   i. Por medio de la clase *MyTask*, el método *doInBackground(...)* lanza el método *verifySignature(...)* para verificar la firma digital del mensaje

   ii. *doInBackground(...)* lanza el método *decodeCVResponse(...)* que devuelve en una lista *String* tanto el mensaje de respuesta, como un mensaje informativo del resultado y el nombre del certificado sobre el que se realizó la acción.

   iii. Se ejecuta el método *onPostExecute(...)* donde se envía dicha lista a la *Actividad6*.

4. *Actividad6* lanza el método *onCreate(...)* donde se recuperan los tres mensajes de tipo *String* y se muestran al usuario por pantalla.

Listado ordenado de procesos para la validación con revocación:

1. *Actividad4* lanza el método *doInBackGround()* de la clase *MyTask* desde donde se lanza el método create*CVRequest*. Este genera la estructura *CVRequest* y la encapsula para su envío.

2. La clase *ServerJ* se mantiene a la escucha por el socket. Recibe la petición de validación e inmediatamente ejecuta los siguientes procesos:

   i. *ServerJ* lanza el método *sendCVResponseToClient(...)* donde se decodifica el mensaje de solicitud

   ii. *sendCVResponseToClient(...)* lanza el método *createDPV()* para la construcción de la ruta y validación de esta.

   iii. *sendCVResponseToClient(...)* lanza el método *verifyCertificateCRLs()* para la comprobación de revocación del certificado.

      a. *sendCVResponseToClient(...)* lanza a su vez los métodos *getCrlDistributionPoints(...)* y *downloadCrl(...)* para la correcta ejecución de este.

   iv. *sendCVResponseToClient(...)* lanza el método *createCVResponse()* para crear el mensaje de respuesta.

   v. *sendValPolResponseToClient(...)* lanza el método *createSignedContentInfo(...)* para firmar digitalmente el mensaje.

   vi. *sendCVResponseToClient(...)* envía dicho mensaje encapsulado.

3. *Actividad4* recibe el mensaje de respuesta por el socket e inmediatamente ejecuta los siguientes procesos:

i. Por medio de la clase *MyTask*, el método *doInBackground(...)* lanza el método *verifySignature(...)* para verificar la firma digital del mensaje

ii. *doInBackground(...)* lanza el método *decodeCVResponse(...)* que devuelve en una lista *String* tanto el mensaje de respuesta, como un mensaje informativo del resultado y el nombre del certificado sobre el que se realizó la acción.

iii. Se ejecuta el método *onPostExecute(...)* donde se envía dicha lista a la *Actividad6*.

4. *Actividad6* lanza el método *onCreate(...)* donde se recuperan los tres mensajes de tipo *String* y se muestran al usuario por pantalla.

4.3.3 Implementación del módulo repositorio de certificados

El módulo del repositorio o almacén de certificados es aquel cuya función es la configuración del almacén interno de certificados digitales correspondiente a la aplicación. Las tareas principales que implementa este módulo, son aquellas relacionadas con el mostrado, exportación o borrado de certificados del almacén.

Dicho módulo al ser íntegramente implementado en la parte del sistema correspondiente al cliente y al no depender de otro tipo de estructuras, únicamente está formado por *Actividad_Local_Cert_Store*. En esta actividad se encuentran cada uno de los métodos cuya función es realizar alguna de las tareas del módulo. En la siguiente **Figura 15** se muestra dicha actividad y los métodos principales contenidos en ella.

Más concretamente, la funcionalidad principal del módulo se divide por un lado en la parte encargada de la exportación de certificados digitales desde la tarjeta SD del dispositivo hasta el almacén interno de la aplicación y por otro lado, en la parte encargada del borrado completo o parcial de los certificados almacenados.

Sin embargo, otra de las tareas realizadas por este módulo de la aplicación, es el mostrado de los certificados disponibles en el repositorio.

Por último, cabe mencionar que dada la necesidad de su uso por todas las funcionalidades anteriormente mencionadas y dada la conexión de este módulo con el módulo criptográfico, otra tarea necesaria a realizar por esta parte es la de búsqueda y listado de los certificados disponibles en un almacén, bien el repositorio externo o bien en un carpeta del repositorio externo. Esto es, que la funcionalidad descrita en la siguiente sección es implementada tanto para el listado de certificados ya contenidos en la aplicación como para certificados contenidos en carpetas de la tarjeta SD del dispositivo.

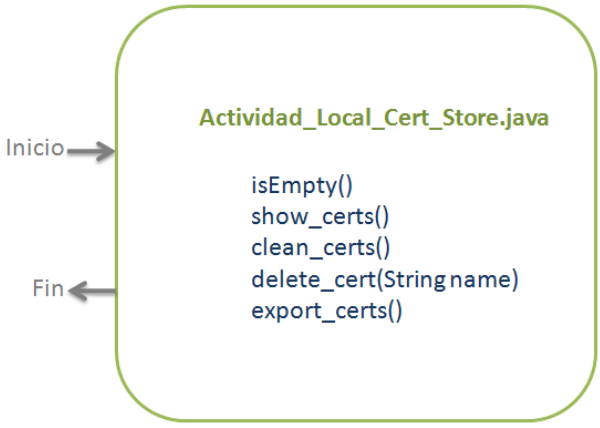

Figura 15. Actividad del módulo repositorio de certificados

A continuación se detallan las lógicas empleadas para las tareas mencionadas anteriormente.

4.3.3.1 Lógica de búsqueda y listado de certificados

En este apartado se explica cómo implementa el sistema la funcionalidad de búsqueda de certificados y su posterior almacenamiento en una lista. Este proceso es esencial para ser ejecutado previamente a la ejecución del resto de funcionalidad de este módulo. Es decir, la búsqueda y listado de los certificados disponibles en una carpeta del dispositivo móvil se aplica antes de realizar una exportación o previo a la realización de un borrado.

El procedimiento general para la obtención de dicha lista de certificados es el recogido en el siguiente diagrama de flujo de la **Figura 16**.

La ejecución del código encargado de la realización de dicha funcionalidad de acuerdo al diagrama de flujo mostrado posteriormente se realiza para los métodos: *show_certs(),clean_certs(), delete_cert(String name)y export_certs()*, que implementan las funcionalidades de mostrado de certificados, borrado total del almacén, borrado selectivo del almacén y exportación de certificados respectivamente. La lógica de dichas funciones se encuentran detalladas en secciones sucesivas.

Por lo tanto, dicho código es considerado como código auxiliar cuya ejecución se repite con anterioridad a la ejecución de código que implementa las funcionalidades principales del módulo correspondiente al almacén de certificados.

El funcionamiento básico, de acuerdo al diagrama mostrado a continuación, es el de comprobación de la disponibilidad de certificados en las carpetas que corresponda y el posterior almacenamiento del nombre del certificado en una lista de tipo: *List<String> listStored = new ArrayList<String>()*, es decir, una lista cuyas entradas son objetos *String* con el nombre del certificado y su extensión.

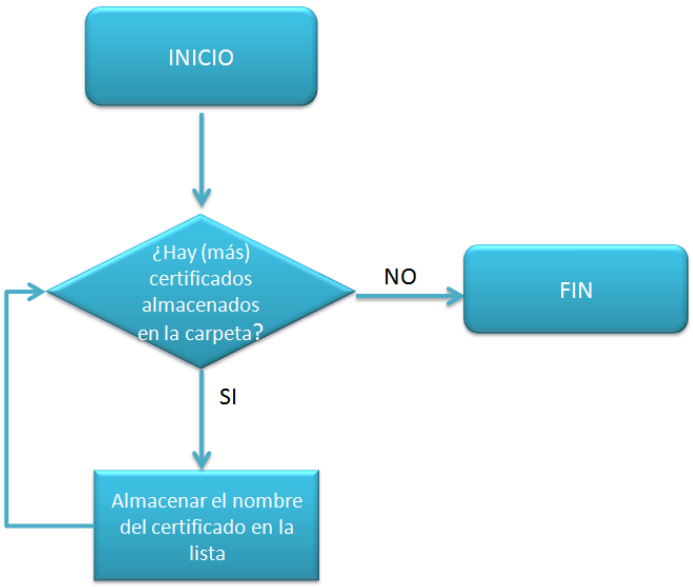

Figura 16. Diagrama de flujo para la búsqueda y listado de certificados

En el caso de la lógica de exportación de certificados, dicho código se ejecuta dos veces. Esto es debido a la necesidad de conocer bien por un lado, los certificados que ya se encuentran almacenados en la memoria de la aplicación (repositorio interno) y por otro lado conocer los certificados que el usuario desea exportar con el fin de que no haya duplicados. Para el resto de métodos, es decir, *show_certs(), clean_certs() y delete_cert(String name*) la ejecución de esta lógica solo se realiza una vez.

4.3.3.2 Lógica de exportación de certificados

En este apartado se detalla como el sistema implementa la exportación de certificados digitales siendo esta una de las funciones básicas del módulo de certificados. La funcionalidad implica el copiado de los certificados digitales que el usuario considera de confianza, desde el almacén externo del dispositivo móvil hacia el almacén interno de la aplicación.

El procedimiento general para dicha exportación es el recogido en el siguiente diagrama de flujo de la **Figura 17.**

El método encargado de la realización de las tareas mostradas en el diagrama anterior es el método *export_certs().* La tarea sobre la que se basa el código implementado en dicho método es la de copiar los certificados que el usuario considere de confianza en el almacén interno de la aplicación y cuya función es la de contribuir a la construcción de la ruta de certificación en base a la disponibilidad de dichos certificados.

Para ello se dispone de una carpeta origen denominada *certificados_confianza* creada en el almacenamiento externo de la tarjeta SD del dispositivo móvil y una carpeta destino, que no es más que la memoria interna de la aplicación.

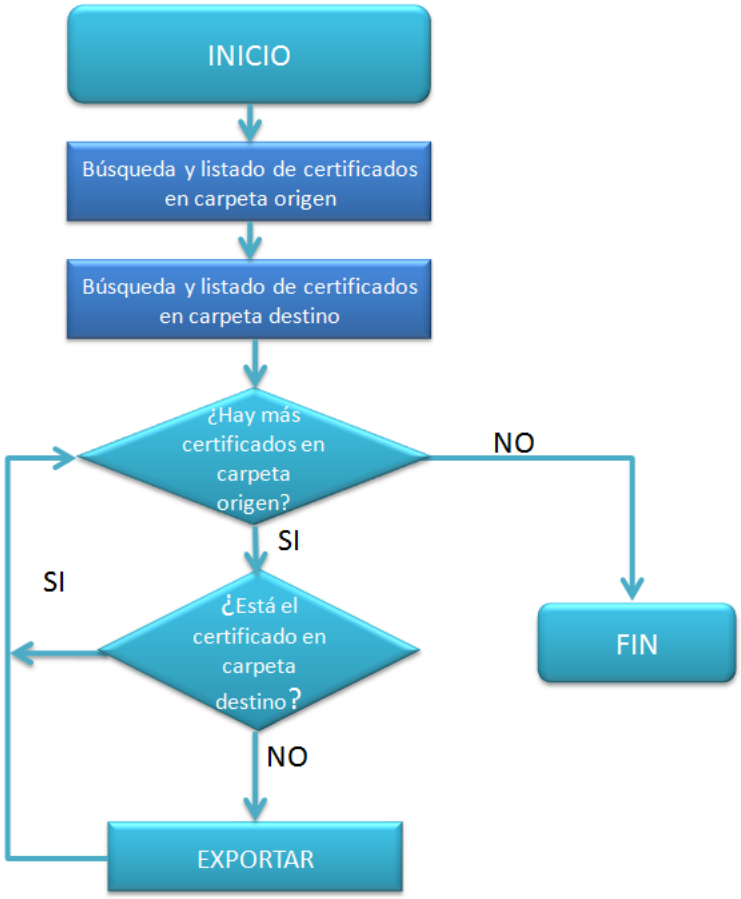

Figura 17. Diagrama de flujo para la exportación de certificados

Por lo tanto, siguiendo los pasos marcados por el diagrama expuesto anteriormente, es necesaria la ejecución del código referente a la búsqueda y posterior listado de certificados digitales explicado en la sección 4.3.3.1 por partida doble. Es necesario conocer el nombre de los certificados a exportar y el nombre de los certificados ya almacenados en la aplicación en caso de que los hubiese para evitar copias duplicadas.

De manera recursiva se procede a comprobar que el nombre del certificado a exportar no se encuentre contenido en la lista del repositorio interno de la aplicación para proceder a la exportación. En caso de que los nombres coincidan, el sistema no realiza dicha copia e informa al usuario. La aplicación también informa al usuario en caso de que la carpeta de origen no contenga certificados.

4.3.3.3 Lógica de mostrado de certificados

En este apartado se explica cómo implementa el sistema la funcionalidad correspondiente al mostrado por pantalla de los certificados digitales que se encuentran almacenados en el repositorio interno de la aplicación.

El procedimiento general para mostrar los certificados disponibles en la carpeta relativa al repositorio de la aplicación, se recoge en el siguiente diagrama de flujo de la **Figura 18**.

La secuencia de actividades a realizar por el método *String show_certs()*, que es el método cuyo código ejecuta dicha funcionalidad, sigue una secuencia sencilla de pasos.

El primer paso de dicha secuencia es la ejecución del código para la búsqueda y listado de certificados digitales disponibles en el almacén, tal como se explicó en la sección anterior 4.3.3.1. Posteriormente, dicha lista de certificados es almacenada en un elemento *String* que será enviado a la interfaz del usuario para su mostrado por pantalla al mismo. En caso de que la lista de certificados se encuentre vacía, el elemento *String* que se devuelve a la interfaz contiene un mensaje de advertencia debido a la inexistencia de certificados de confianza en el repositorio.

Figura 18. Diagrama de flujo para el mostrado de certificados

4.3.3.4 Lógica de borrado total del almacén

En este apartado se explica cómo implementa el sistema la funcionalidad correspondiente al borrado completo del repositorio de certificados de confianza de la aplicación. Es decir, se especifica cuál es el procedimiento aplicado por eliminar de la memoria de la aplicación todos los certificados, considerados de confianza, previamente almacenados por el usuario.

El procedimiento general para la eliminación de todos los certificados disponibles en la carpeta relativa al repositorio de la aplicación, es el recogido en el siguiente diagrama de flujo de la **Figura 19**.

El método encargado de la implementación de dicha funcionalidad en el módulo es el método *void clean_certs()*. Al igual que para el resto de funcionalidades implementadas, el primer paso a realizar en esta ejecución, es la búsqueda y posterior listado de certificados digitales explicado en la sección 4.3.3.1.

De manera recursiva y hasta que la lista de certificados no se haya vaciado se procederá a eliminación del certificado que se encuentre situado primero en la lista. Una vez ejecutado dicho borrado, se procede de nuevo al listado de los certificados disponibles.

Esta lógica no se ejecutará ninguna vez, si en la primera búsqueda de certificados no se encuentra ninguno y por lo tanto, la lista está vacía.

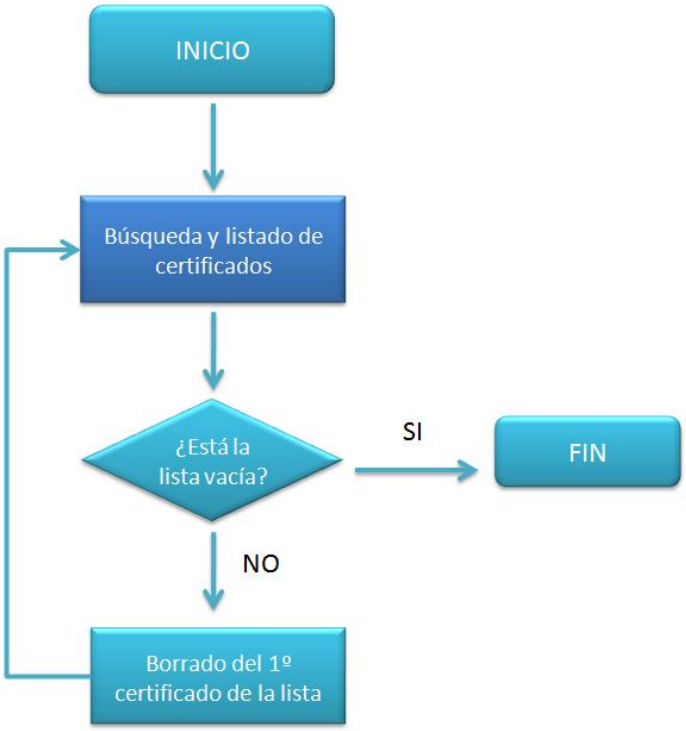

Figura 19. Diagrama de flujo para el borrado total del almacén de certificados

4.3.3.5 Lógica de borrado selectivo del almacén

En este último apartad se explica cómo el sistema implementa la funcionalidad correspondiente al borrado selectivo de certificados del repositorio interno de la aplicación. Es decir, la explicación versa sobre la búsqueda de un certificado concreto y su posterior borrado en caso de que este se encuentre en el repositorio.

El procedimiento general para la eliminación de un único certificado elegido por el usuario es el recogido en el siguiente diagrama de flujo de la **Figura 20.**

El método encargado de la implementación del procedimiento expuesto en la figura anterior es el método*: boolean delete_cert(String name).* Dicho método tiene la finalidad de realizar el borrado de un certificado digital cuyo nombre es introducido por el usuario gracias a la interfaz de la aplicación. Este nombre es recibido por el método *delete_cert(String name)*por parámetro.

Una vez se cuenta con el nombre del certificado que el usuario desea borrar, es necesario al igual que para el resto de funcionalidades, ejecutar el código referente a la

búsqueda y posterior listado de certificados digitales explicado en la sección [4.3.3.1], con el fin de corroborar dicho nombre con los certificados disponibles.

En caso de que el nombre del certificado de usuario no corresponda con el nombre de ninguno de los certificados almacenados o que la aplicación no cuente con certificados almacenados, dicho método no ejecuta ningún borrado y el resultado que devuelve es *false*.

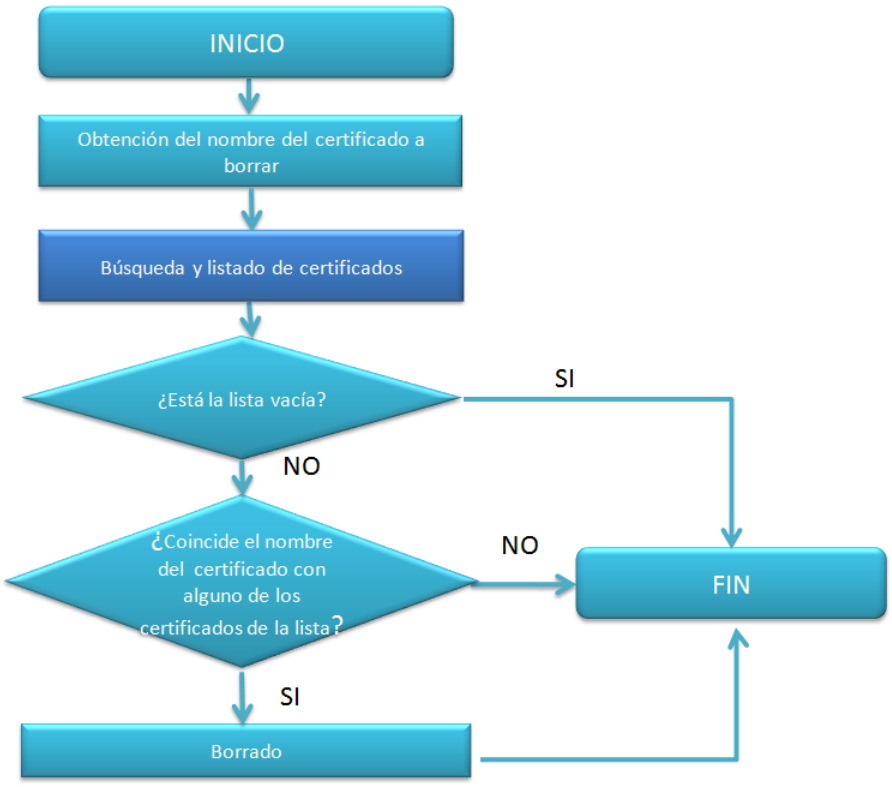

Figura 20. Diagrama de flujo para el borrado selectivo de un certificado

En cambio, si el nombre del certificado de usuario si corresponde con el nombre de alguno de los certificados almacenados en el repositorio, este método procederá a la eliminación de dicho certificado del almacén, tal como el usuario requería.

Capítulo 5

Pruebas

En este capítulo se presentan las pruebas realizadas así como los resultados obtenidos para la comprobación del correcto funcionamiento del proyecto. Para ello, dichas pruebas han sido estructuradas y diseñadas en relación al módulo del proyecto que se desea probar ya que la utilización de este tipo de división permite la detección y solución de problemas de manera aislada.

Las diferentes pruebas realizadas se encuentran, a continuación, en tablas que cuentan con un identificador de la prueba, una descripción de la misma y el resultado de esta. Los códigos de identificación de las pruebas son los mostrados en la siguiente tabla.

Identificador de la prueba	Tipo de prueba
IU	Interfaz de usuario
M-PS	Módulo Política del Servidor
M-AC	Módulo Almacén de Certificados
M-DPD	Sub-Modulo DPD
M-DPV	Sub-Módulo DPV

Tabla 12. Códigos de identificación de pruebas

5.1 Preparación entorno de pruebas

El escenario con el que se cuenta para la comprobación de dichas pruebas antes de la prueba final en un dispositivo móvil, es un emulador AVD (*Android Virtual Device*). Este dispositivo virtual de Android permite simular desde Eclipse cualquier dispositivo que cuente con sistema operativo Android. De esta manera, es posible probar la aplicación simulando los escenarios que desempeñarían el funcionamiento completo del sistema. Su configuración se encuentra detallada en la sección B.2.3.

Para la realización de las pruebas correspondientes al módulo política del servidor, al módulo almacén de certificados y al módulo criptográfico fue necesario el almacenamiento de certificados digitales. Dichos certificados han sido creados programáticamente en Java y almacenados en distintos lugares.

Para las pruebas del módulo política del servidor y del módulo criptográfico, dado que se realiza una conexión entre el cliente y el servidor, y los mensajes de este último van firmados digitalmente, fue necesario el almacenamiento de un certificado para el servidor en la memoria del PC desde donde se lanzó este.

Además para las pruebas del módulo criptográfico y del almacén de certificados fue necesario el almacenamiento de certificados de usuario en una carpeta llamada *certificados_usuario* y el almacenamiento de certificados de confianza, es decir, raíz e intermedios en una carpeta denominada *certificados_confianza*. Ambas carpetas deben estar situadas en el almacenamiento externo del dispositivo, tarjeta SD.

5.2 Pruebas de la interfaz de usuario

En el proceso de pruebas de la interfaz de usuario, se pretende comprobar que tanto las distintas vistas de la aplicación como sus elementos cumplen sus funciones de manera adecuada. Asimismo, se examina el funcionamiento de otro tipos de eventos tales como la recepción de mensajes de error o los diálogos de selección.

Dichas pruebas se encuentra recogidas en la siguiente **Tabla 13**.

Ident.	Descripción	Resultado esperado	Estado
IU.1	Cambios entre las diferentes vistas	La navegación por las pantallas de la aplicación responde al modelo diseñado	OK
IU.2	Campos editables	Recogen la información introducida por el usuario. Muestran el valor actual y permiten su modificación	OK
IU.3	Menús de selección	Se capturan y ejecutan los elementos de selección	OK

IU.4	Carga de datos en las vistas	La información resultante es cargada correctamente	OK
IU.5	Mensajes de aviso	Es lanzando al detectarse un error de configuración. Desaparece al pulsar OK	OK

Tabla 13. Pruebas de la interfaz de usuario

5.3 Pruebas del módulo política del servidor

En el proceso de pruebas correspondiente al módulo de la política del servidor, se pretende comprobar el correcto funcionamiento e intercambio de mensajes entre el cliente y el servidor con motivo de la consulta de dicha política. Se examina por un lado, la correcta recepción de la solicitud en el servidor y la correcta recepción de la respuesta en el cliente.

Dichas pruebas se pueden consultar en la siguiente **Tabla 14**.

Ident.	Descripción	Resultado esperado	Estado
M-PS.1	Envío de solicitud de política con los campos correspondientes al elemento Nonce vacíos	El valor Nonce del mensaje de petición tiene como valor 0000	OK
M-PS.2	Envío de solicitud de política con los todos los campos correspondientes al elemento Nonce rellenados	El valor Nonce del mensaje de petición contiene el valor introducido por el usuario	OK
M-PS.3	Envío de solicitud de política con algún campos correspondiente al elemento Nonce vacío	El valor Nonce del mensaje de petición contiene el valor 0 en aquellos campos no rellenados por el usuario	OK
M-PS.4	Envío del mensaje de respuesta de política estandar desde el servidor	La operación se realiza correctamente. Mensaje creado y enviado de manera satisfactoria	OK

Tabla 14. Pruebas del módulo política del servidor

5.4 Pruebas del almacén de certificados

En el proceso de pruebas del almacén de certificados se llevan a cabo pruebas relacionadas con la configuración del repositorio interno de certificados de la aplicación. Las pruebas tratan de corroborar la correcta exportación y borrado de certificados en dicho almacén. Así como, de la búsqueda y listado de los certificados disponibles en la memoria interna.

Dichas pruebas se encuentra recogidas en la siguiente **Tabla 15.**

Ident.	Descripción	Resultado esperado	Estado
M-AC.1	Exportación de los certificados de confianza del usuario desde el almacenamiento externo del dispositivo móvil al almacenamiento interno de la aplicación si la carpeta *certificados_confianza* no existe. El almacén interno se encuentra vacío o con certificados.	La aplicación informa al usuario de la necesidad de creación o corrección del nombre de la carpeta *certificados_confianza*	OK
M-AC.2	Exportación de los certificados de confianza del usuario desde el almacenamiento externo del dispositivo móvil al almacenamiento interno de la aplicación si la carpeta *certificados_confianza* está vacía. El almacén interno se encuentra vacío o con certificados.	La aplicación informa al usuario del contenido nulo de la carpeta *certificados_confianza*	OK
M-AC.3	Exportación de los certificados de confianza del usuario desde el almacenamiento externo del dispositivo móvil al almacenamiento interno de la aplicación si la carpeta *certificados_confianza* contiene certificados. El almacén interno se encuentra vacío.	La operación se realiza correctamente. Los certificados se copian en el almacenamiento interno de la aplicación.	OK
M-AC.4	Exportación de los certificados de confianza del usuario desde el almacenamiento externo del dispositivo móvil al almacenamiento interno de la aplicación si la carpeta *certificados_confianza* contiene certificados. El almacén interno contiende certificados con el mismo nombre que algunos de los certificados seleccionados para exportar.	La aplicación informa al usuario del duplicado de los nombres de certificados en ambos almacenamientos y no realiza la exportación de aquellos duplicados.	OK
M-AC.5	Borrado de todos los certificados de confianza del almacenamiento interno de la aplicación. El almacén interno se encuentra vacío.	El almacén interno permanece vacío. La aplicación informa al usuario del estado vacío del almacén interno	OK
M-AC.6	Borrado de todos los certificados de confianza del almacenamiento interno de la aplicación. El almacén interno contiene certificados.	El almacén interno se vacía de certificados. La aplicación informa al usuario del estado vacío del almacén interno.	OK
M-AC.7	Borrado de un certificado digital cuyo nombre es introducido por el usuario. El almacén interno se encuentra vacío.	La aplicación informa al usuario del estado vacío del almacén interno.	OK
M.AC-8	Borrado de un certificado digital cuyo nombre es introducido por el usuario. El certificado se encuentra en el almacén interno.	El certificado seleccionado se borra del almacén interno. Se muestra al usuario la lista actualizada de certificados disponibles	OK
M-AC-9	Borrado de un certificado digital cuyo nombre es introducido por el usuario. El certificado no se encuentra en el almacén interno.	La aplicación informa al usuario de la no existencia de un certificado con ese nombre. No se realiza operación sobre el almacén.	OK

Tabla 15. Pruebas del módulo almacén de certificados

5.5 Pruebas del módulo criptográfico

5.5.1 Pruebas del sub-módulo DPD

En el proceso de pruebas del sub-módulo DPD, se llevan a cabo pruebas relacionadas con la construcción de la ruta de certificación en base a un certificado de usuario y varios certificados de confianza almacenados previamente en el almacén interno de la aplicación.

El conjunto de las pruebas realizadas en este módulo se encuentran recogidas en la siguiente **Tabla 16**.

Ident.	Descripción	Resultado esperado	Estado
M-DPD.1	Construcción de la ruta de certificación de un certificado **válido**. El almacén interno se encuentra vacío	No se puede calcular la ruta	OK
M-DPD.2	Construcción de la ruta de certificación de un certificado **inválido**. El almacén interno se encuentra vacío	No se puede calcular la ruta	OK
M-DPD.3	Construcción de la ruta de certificación de un certificado **válido** cuya ruta **SI** puede ser construida con los certificados contenidos en el almacén interno.	**Construcción correcta de la ruta de certificación**	OK
M-DPD.4	Construcción de la ruta de certificación de un certificado **inválido** cuya ruta **SI** puede ser construida con los certificados contenidos en el almacén interno	**Construcción correcta de la ruta de certificación**	OK
M-DPD.5	Construcción de la ruta de certificación de un certificado **válido** cuya ruta **NO** puede ser construida con los certificados contenidos en el almacén interno	No se puede calcular la ruta	OK
M-DPD.6	Construcción de la ruta de certificación de un certificado **inválido** cuya ruta **NO** puede ser construida con los certificados contenidos en el almacén interno	No se puede calcular la ruta	OK

Tabla 16. Pruebas del sub-módulo DPD

5.5.2 Pruebas del sub-módulo DPV

En el proceso de pruebas del sub-módulo DPV, correspondiente a la validación de un certificado y su correspondiente ruta de certificación, se llevan a cabo pruebas relacionadas con la validación tanto del certificado de usuario en cuestión como de todos los certificados que conforman la ruta. Las pruebas tratan de corroborar la validación en caso de que sea posible previo cálculo de dicha ruta de certificación. Se contempla, dentro de la batería de pruebas, que el usuario requiera la validación con y sin comprobación de revocación.

La batería de pruebas realizadas para este módulo DPV se recogen en la siguiente **Tabla 17**.

Ident.	Descripción	Resultado esperado	Estado
M-DPV.1	Validación, sin comprobación de revocación, de un certificado **válido**. El almacén interno se encuentra vacío	La validación no se puede realizar	OK
M-DPV.2	Validación, sin comprobación de revocación, de un certificado **inválido**. El almacén interno se encuentra vacío	La validación no se puede realizar	OK
M-DPV.3	Validación, sin comprobación de revocación, de un certificado **válido** cuya ruta SI puede ser construida y validada con los certificados contenidos en el almacén interno	**La validación se ha realizado correctamente**	OK
M-DPV.4	Validación, sin comprobación de revocación, de un certificado **inválido** cuya ruta SI puede ser construida y validada con los certificados contenidos en el almacén interno	La validación no se puede realizar	OK
M-DPV.5	Validación, sin comprobación de revocación, de un certificado **válido** cuya ruta NO puede ser construida y validada con los certificados contenidos en el almacén interno	La validación no se puede realizar	OK
M-DPV.6	Validación, sin comprobación de revocación, de un certificado **inválido** cuya ruta NO puede ser construida y validada con los certificados contenidos en el almacén interno	La validación no se puede realizar	OK
M-DPV.7	Validación, con comprobación de revocación, de un certificado **válido** que no tiene asociados distribution points. El almacén interno se encuentra vacío	La validación no se puede realizar	OK
M-DPV.8	Validación, con comprobación de revocación, de un certificado **inválido** que no tiene asociados distribution points. El almacén interno se encuentra vacío	La validación no se puede realizar	OK
M-DPV.9	Validación, con comprobación de revocación, de un certificado **válido** cuya ruta NO puede ser construida con los certificados contenidos en el almacén interno y que NO tiene asociados distribution points	La validación no se puede realizar	OK
M-DPV.10	Validación, con comprobación de revocación, de un certificado **inválido** cuya ruta NO puede ser construida con los certificados contenidos en el almacén interno y que NO tiene asociados distribution points	La validación no se puede realizar	OK
M-DPV.11	Validación, con comprobación de revocación, de un certificado **válido** cuya ruta SI puede ser construida y/o validada con los certificados contenidos en el almacén interno y que NO tiene asociados distribution points	La validación no se puede realizar. El sistema no cuenta con fuente de revocación del certificado	OK
M-DPV.12	Validación, con comprobación de revocación, de un certificado **inválido** cuya ruta SI puede ser construida con los certificados contenidos en el almacén interno y que NO tiene asociados distribution points	La validación no se puede realizar	OK

Tabla 17. Pruebas del sub-módulo DVP

Capítulo 6

Histórico del proyecto

6.1 Fases

En este capítulo seis se recogen las diferentes etapas llevadas a cabo para la realización del proyecto. Para un mejor entendimiento, se encuentran explicados para cada una de las fases, los objetivos y tareas realizadas así como los problemas encontrados en el camino.

Asimismo a lo largo de este análisis del histórico del proyecto se encuentran reflejadas las diferentes alternativas de diseño consideradas y la justificación del diseño final implementado para el sistema.

Este proyecto se encuentra dividido en cuatro fases. La primera de ellas es la fase de definición de objetivos, posteriormente se encuentra la fase de planteamiento inicial, después la fase de implementación del sistema y por último la de documentación. Cada una de ellas, como se mencionó anteriormente, se encuentra desglosada en las diferentes tareas realizadas, los problemas surgidos y los resultados finales obtenidos.

No obstante, las pruebas realizadas aunque en este capítulo no se contemplen, debido a que se encuentran definidas en el capítulo 5, se llevaron a cabo en paralelo al desarrollo de las distintas fases.

Previo a la explicación en detalle de cada una de ellas, en la siguiente **Figura 21** se encuentra esbozado un gráfico orientativo sobre las cuatro fases que compusieron el desarrollo del proyecto y una breve descripción de cada una de ellas.

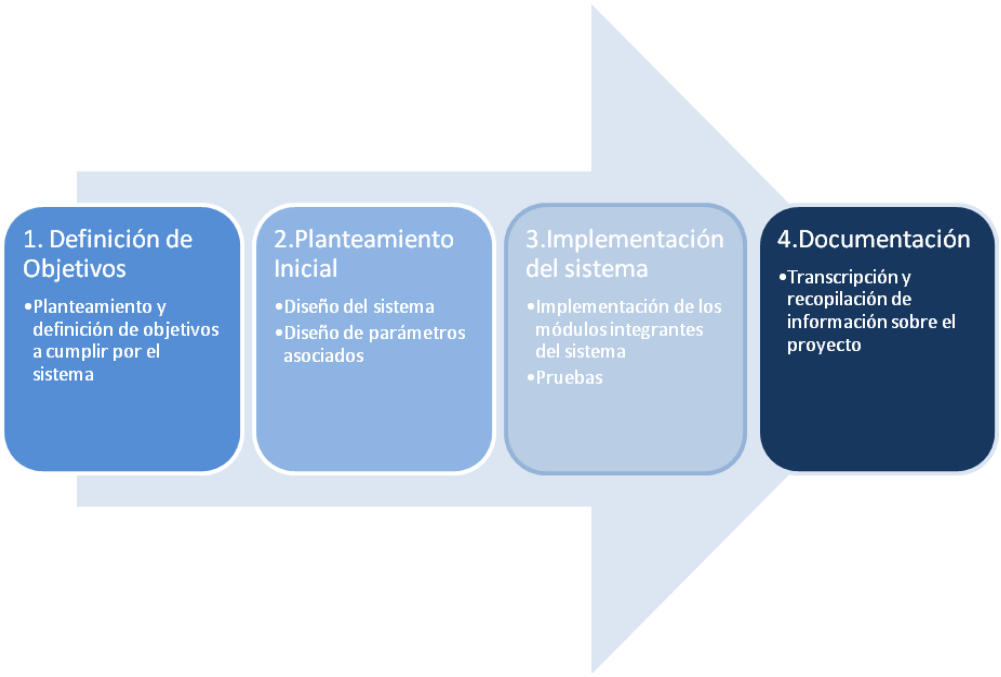

Figura 21. Gráfico orientativo de las fases del proyecto

6.1.1 Fase I. Definición de objetivos

- **Tareas realizadas**

  El planteamiento de partida sobre el que trabajar fue, en primera instancia, la validación de certificados digitales desde un terminal móvil. Concretamente se planteó la implementación del protocolo SCVP, definido en la RFC 5055 [5].

  En base a esta idea preliminar se definieron una serie de objetivos sobre los que se sustentaba el desarrollo del proyecto. Se eligió el uso del certificado digital *X.509* como formato de certificado más utilizado, el protocolo HTTP como protocolo de aplicación para el desarrollo y el uso del sistema operativo Android como sistema operativo móvil.

  Una vez definidos los objetivos fundamentales, se inició una fase de documentación, para el estudio de la viabilidad del proyecto, así como de las herramientas de desarrollo necesarias para este y el entorno de trabajo.

  Para comprender el funcionamiento del protocolo SCVP, que en definitiva es el pilar fundamental sobre el que descansa el proyecto, fue necesario el estudio detallado de la RFC 5055 [5]. Es en este documento donde se encuentran detallados los procedimientos de mensajería, estructuras de datos y protocolos de aplicación. Como documento complementario a este y

necesario para el entendimiento de la infraestructura de clave pública así como del certificado digital *X.509* fue necesario recurrir a la RFC 5280 [2].

En lo referido al manejo de la criptografía de clave pública en lenguaje de programación Java, se examinaron algunos libros, siendo el principal [16] con el fin de extraer ideas y explorar diferentes vías para su implementación.

Y por último, con el objeto de comprender las herramientas de programación en Android, se instaló el IDE de Eclipse junto con el SDK de Android tal como se detalla en B.2.3. Asimismo, como iniciación al diseño de una aplicación en Android, se consultó la página web de desarrolladores [17] para obtener una primera toma de contacto con el mecanismo de diseño y programación.

Dentro de esta primera fase, se encuentran enmarcadas las primeras pruebas cuya finalidad era la de adquirir soltura con el entorno. En ellas se analizaron y ejecutaron varios ejemplos de programas en Android utilizando un AVD, cuya configuración se encuentra detallada en la sección B.2.3.

- **Problemas**

  El primer problema planteado durante esta fase versa sobre la elección del lenguaje de programación utilizado para las funcionalidades desempeñadas en el lado del servidor. En un principio se pensó en realizar el desarrollo del servidor en lenguaje C pero esto hacía necesaria la programación en dos lenguajes diferentes ya que la plataforma Android, utilizada en el lado del cliente, se desarrolla en lenguaje Java.

- **Resultados**

  Como resultado de esta primera fase, se descartó el uso del lenguaje de programación C y en su lugar se decidió utilizar el lenguaje Java para el desarrollo completo del proyecto. Aunque C había sido elegido en principio, por su facilidad de uso con certificados digitales, Java resultó de gran utilidad gracias a sus librerías nativas para la parte criptográfica y a la API de *Bouncy Castle*, previamente mencionada en 2.4.2 . Además este último lenguaje proporcionaba una comunicación más ágil y unificada entre el cliente y el servidor.

6.1.2 Fase II. Planteamiento inicial

- **Tareas realizadas**

  Una vez sentadas las directrices más básicas a utilizar en el diseño del proyecto, siempre apoyadas sobre las instrucciones marcadas por el protocolo SCVP, se procedió a marcar la ruta inicial del desarrollo para el sistema.

  Se estableció como definitivo el uso del lenguaje de programación Java para el desarrollo de las actividades integradas en el servidor y el uso de la

plataforma Android para el desarrollo de las actividades integradas en el cliente.

Por lo tanto, como propuesta inicial del proyecto se planteó un sistema basado en una arquitectura cliente-servidor, donde el primero de ellos fuera una aplicación móvil diseñada en sistema operativo Android y el segundo una plataforma servidor diseñada en Java. El cliente, era por tanto el poseedor de los certificados digitales y solicitante de servicios y el servidor el encargado de prestar dichos servicios y dar respuesta a la petición del cliente.

De igual forma, como parte de esta fase de planteamiento inicial, se concretaron los parámetros finales relacionados con el intercambio de mensajes imprescindible entre los dos elementos base del sistema. Otro aspecto relevante de esta fase del proyecto, fue el estudio y definición asociado a las funciones criptográficas que el servidor debía implementar, tales como la metodología de construcción de una ruta de certificación o validación de dicha ruta.

- **Problemas**

El inconveniente más relevante que surgió durante esta segunda fase tuvo que ver con el almacenamiento y procesado de certificados digitales por parte del usuario en la aplicación móvil.

El acceso al almacén de certificados nativos de Android para la extracción o manipulación de certificados por parte de la aplicación móvil resultó inviable. Este tipo de acciones, imprescindibles para el desarrollo del proyecto, no se encontraban disponibles dentro de la plataforma.

El otro obstáculo destacado a salvar durante el planteamiento del proyecto, fue el estricto protocolo en cuanto a estructuras de datos se refiere, establecido para la conexión y envío seguro de datos entre el cliente y el servidor.

- **Resultados**

Dada la relevancia de los problemas surgidos durante esta etapa del proyecto y su impacto en la correcta ejecución del proyecto, resultó de vital importancia subsanar dichas complicaciones.

En primer lugar, la manera de salvar el problema encontrado en cuanto al almacenamiento de certificados digitales en la aplicación móvil, fue el diseño de un repositorio interno de certificados desde el que tanto el usuario como la aplicación pudieran gestionar dichos certificados de una manera eficiente.

Respecto al estricto y cargado nivel de protocolo establecido sobre los mensajes entre ambas partes, fue determinante el estudio de documentación adicional sobre el envío de mensajes firmados digitalmente o estructuras predefinidas en notación ASN.1 en la API de BouncyCastle. Tanto el procesado de información en cada mensaje como la traducción desde la

notación ASN.1 a notación Java supusieron puntos determinantes de estudio para la definición de dicha fase.

En definitiva, el funcionamiento del programa, como ya ha sido introducido previamente, estaba basado en un sistema cuyo cliente solicita un servicio al servidor en base a un protocolo previamente definido.

6.1.3 Fase III. Implementación del sistema

Una vez definidas las bases que sustentaban el diseño completo del sistema, se procedió a realizar la implementación de este. Dicha implementación, se realizó en torno a una división previa de sistema por módulos, tal como se explica en la sección 3.3. Es importante resaltar, que para el módulo que comprende la interfaz de usuario, la implementación ha sido realizada en paralelo a la de los otros módulos.

6.1.3.1 Implementación del módulo política del servidor

- **Tareas realizadas**

  En primer lugar se procedió a desarrollar la parte correspondiente al intercambio de mensajes para la consulta y respuesta de la política del servidor. Esta parte comprendía por un lado la creación de la estructura del mensaje de solicitud del cliente y su posterior envío al servidor. Posteriormente se procedió al procesado de dicha solicitud y creación del mensaje de respuesta, cuyo cuerpo contenía la política, para su posterior envío al cliente. Por último fue necesario el procesado de dicho mensaje de respuesta en el cliente para poder ser mostrado al usuario por medio de la interfaz de la aplicación.

- **Problemas**

  Los problemas derivados de esta implementación surgieron principalmente por la conexión inicial para el tráfico de datos entre la aplicación cliente y el servidor. La rutina de comunicación y el protocolo a seguir fueron los principales obstáculos a eludir en la primera etapa de la fase de implementación

- **Resultados**

  Tras la finalización de la implementación del módulo, se obtiene un intercambio de mensajes correcto entre el cliente y el servidor cuyo objeto es la consulta de la política estipulada por el servidor para la validación de certificados.

  La creación y envío, tanto del mensaje de solicitud como del de respuesta, se realizan correctamente siguiendo las bases marcadas por el protocolo.

Asimismo en esta primera parte de la implementación se ejecuta y comprueba la correcta exposición de dicha política en la interfaz del usuario en la aplicación móvil.

6.1.3.2 Implementación del módulo almacén de certificados

- **Tareas realizadas**

En segundo lugar, una vez establecida una primera conexión entre el cliente y el servidor y realizado el envío de mensajes básicos, se procedió a la implementación del repositorio o almacén local de certificados en el cliente.

En cuanto a las actividades llevadas a cabo para el desarrollo de esta sub-fase figuran la implementación de la funcionalidad de exportación de certificados digitales desde el sistema de almacenamiento externo y el borrado de elementos acumulados en el repositorio. También se implementó la funcionalidad de lectura de dicho almacén para su posterior muestra al usuario de los elementos disponibles en este.

Este módulo se implementó con anterioridad al módulo criptográfico debido a la necesidad de utilización de este último de los certificados almacenados.

- **Problemas**

Los inconvenientes surgidos en esta parte de la implementación tuvieron como principal punto en común el copiado y pegado de certificados desde un punto del dispositivo móvil a otro. Es decir, fue la exportación de mensajes desde el almacenamiento externo en la tarjeta SD del dispositivo, al almacenamiento interno de la aplicación la que provocó los principales problemas de implementación en esta parte.

- **Resultados**

Con la realización de este módulo quedó establecido en el cliente un lugar seguro de almacenamiento de sus certificados de confianza, ya que el almacenamiento interno por parte de la aplicación hacía que solo dicha aplicación pudiera acceder a ese contenido.

Por lo tanto, desde este punto la aplicación contó con un repositorio de certificados del usuario no accesible desde otra aplicaciones y que permitía a este tanto añadir como borrar certificados. Un repositorio local adaptado al usuario.

6.1.3.3 Implementación del módulo criptográfico

- **Tareas realizadas**

Como última pare de la implementación y siendo esta la más importante dentro del cómputo global del proyecto, se procedió al desarrollo de las actividades concernientes al módulo criptográfico. Es decir, se

implementaron las funciones tanto de construcción de la ruta de certificación como la validación de esta.

Primero, fue implementada la funcionalidad correspondiente a la construcción de la ruta de certificados. Dicha funcionalidad comprendía tanto la creación del mensaje de petición del cliente al servidor, con los campos correspondientes a dicha funcionalidad, el desarrollo de la función que generara dicha ruta y la creación del mensaje de respuesta conteniendo el resultado de la petición.

Una vez, creada la funcionalidad de la ruta se procedió a la implementación del sub-módulo correspondiente a la validación de dicha ruta. Esta implementación, fue a su vez, realizada en dos partes. En la primera parte se desarrolló la validación sin comprobación de revocación de los certificados y en la segunda parte se procedió finalmente a la implementación de la validación teniendo en cuenta la comprobación de revocación.

- **Problemas**

  En cuanto a la parte de implementación del sistema, es esta última fase por ser la más extensa y la más primordial, la que más problemas en cuanto a desarrollo provocó. La utilización de los métodos, estructuras y APIs de criptografía de Java, y en concreto de Bouncy Castle, hicieron de esta parte la más difícil de implementar.

  Tanto el procesado de los certificados digitales, como la obtención de sus campos y rutas o la utilización de la listas de revocación fueron los principales inconvenientes surgidos durante la implementación de dicha parte.

- **Resultados**

  Con la realización de este último módulo, quedó integrada la parte criptográfica fundamental del sistema. Desde este punto, el sistema era capaz de realizar tanto labores de construcción de la ruta como de validación de un certificado en concreto suministrado por el cliente.

  Una vez finalizada la fase III de implementación del sistema, quedaron integradas las funcionalidades de consulta/recepción de la política del servidor, la funcionalidad referente a la configuración del repositorio de certificados y las funcionalidades criptográficas fundamentales del sistema. En esta parte de la implementación, se realizó a su vez, la mayor parte de implementación de la interfaz del usuario tanto para la petición como procesado de la respuesta, así como de la batería de pruebas mayoritarias del sistema.

6.1.4 Fase IV. Documentación

Esta última fase consiste en la documentación de todo el trabajo realizado a lo largo del proyecto y que se encuentra recogida en esta memoria.

6.2 Resumen

Como suplemento a la información detallada en las secciones anteriores y con el fin de proporcionar una visión global del tiempo invertido para las fases previamente descritas, se detalla en la siguiente **Tabla 18** una planificación temporal. Cabe mencionar que la realización del proyecto se realizó entre los meses de Noviembre de 2013 y Septiembre de 2014 a excepción del mes de Enero.

Fase	Descripción	Duración (meses)
Fase I	Definición de objetivos	2
Fase II	Planteamiento inicial	2,5
Fase III	Implementación del sistema	3
Fase IV	Documentación	1,5

Tabla 18. Duración asociada a cada fase del proyecto

Como se puede observar, las primeras fases del proyecto acaparan una gran inversión de tiempo. Tanto el proceso de definición de objetivos como el planteamiento inicial concentraron la mitad del tiempo de la realización del proyecto. Esto es debido a la necesidad de investigación y planteamiento del desarrollo de la aplicación que llevó consigo la generación y tráfico de mucha información y documentación, directamente derivada de la realización de las tareas y fases metodológicas.

No obstante fue la implementación del sistema, parte central del desarrollo, la más larga. En esta fase, la necesidad de consultar documentación adicional la que prolongó la implementación del sistema (Fase III).

Como muestra global del reparto de tiempo entre las distintas fases por las que pasó el proyecto se adjunta la siguiente gráfica en la **Figura 22**.

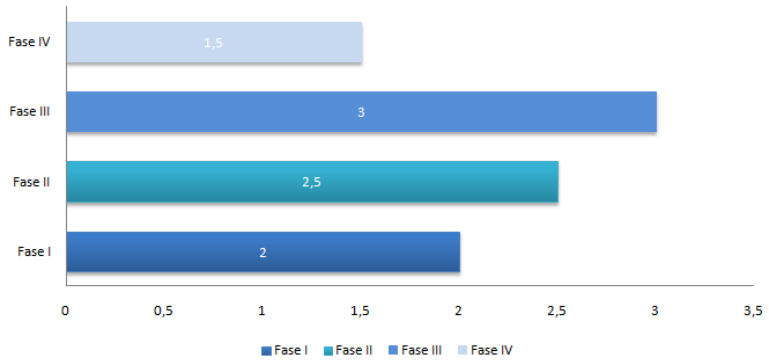

Figura 22. Gráfico de la duración de las etapas del proyecto

Capítulo 7

Conclusiones y líneas futuras

7.1 Conclusiones

Un certificado digital, en la era de Internet, supone un resguardo de identificación en la red. De esta manera permite la realización de gestiones on-line asegurando y preservando la protección de los datos en la comunicación. Debido a la necesidad de adaptarse a las necesidades informáticas de los usuarios, los certificados digitales cada vez obtienen mayor utilidad dentro de los dispositivos móviles.

Bajo estas premisas residían los objetivos fundamentales de este proyecto, basado en ofrecer al usuario una alternativa rápida y sencilla de validación de sus certificados digitales. Este proyecto se focalizaba, en primera instancia, en la unificación de varios procesos y consultas necesarias en un solo sistema que acercara una solución funcional al usuario. En esencia, los objetivos del proyecto han sido cubiertos y abren la puerta a futuras implementaciones abordables más adelante.

Por otro lado, Android es el sistema operativo móvil más utilizado en la actualidad. El hecho de que el diseño de la aplicación se encuentre basado en esta plataforma permite ofrecer una funcionalidad de mayor apertura al mercado.

En definitiva, este proyecto contribuye y agiliza la utilización de los certificados digitales por parte de un usuario móvil con sistema operativo Android mediante el desarrollo de un sistema que permite la construcción y/o validación de los certificados desde el terminal. La aplicación evita la necesidad de realizar múltiples comprobaciones

manuales para corroborar el estado actual del certificado. Además posibilita al usuario el almacenamiento de sus certificados de confianza para futuras comprobaciones.

Por último, la aplicación ha sido desarrollada en un entorno creado a partir de herramientas de *software* libre, lo que permite un fácil acceso a la misma para introducir modificaciones o mejoras de manera asequible.

7.2 Líneas futuras

Como conclusión final del proyecto, se presentan a continuación posibles formas de ampliación o extensión de la funcionalidad del sistema implementado. Estas ampliaciones sugeridas contribuyen a la creación de un sistema más robusto y aportan nuevas alternativas para el cliente.

Las líneas de futuro se han dividido en tres grandes bloques. Por un lado se enmarcan aquellas ampliaciones en cuanto a validación se refiere. En otro bloque quedan encuadradas aquellas mejoras basadas en una ampliación de pruebas. Y por último aquellas mejoras relacionadas con la interfaz del cliente.

7.2.1 Bloque I: Mejoras sistema validación

- **Soporte de validación de varios certificados digitales simultáneamente**
  La aplicación implementada ofrece al cliente la opción de construcción y/o validación de la ruta de certificación de un único certificado digital por cada petición. Es decir, si el cliente desea validar varios certificados, debe realizar el mismo número de peticiones al servidor como certificados desee comprobar. Por esta razón, sería considerable ofrecer al cliente la opción de realizar la construcción y/o validación de la ruta de varios certificados digitales simultáneamente. Este proceso agilizaría estos mecanismos al usuario de la aplicación. De esta manera, el usuario recibiría un resultado individualizado para cada uno de los certificados digitales que se encontraran contenidos en la petición.

- **Soporte de validación de certificados digitales de atributo**
  El sistema desarrollado se encuentra diseñado para realizar tareas de construcción de rutas y/o validación sobre certificados de usuario. Por ende, tanto la aplicación móvil cliente como el servidor se encuentran diseñados para funcionar con dichos certificados. No obstante, resultaría fácilmente adaptable al sistema ya implementado la realización de una ampliación de dichas funcionalidades que permitiera la ejecución de dichas tareas para certificados de atributos.

- **OCSP**
  El mecanismo de comprobación de revocación que el sistema implementado utiliza sobre los certificados digitales, está basado en las listas de revocación, o también llamadas CRLs. Dado que este no es el único mecanismo existente

en la actualidad para dicho fin, resultaría interesante incluir en la comprobación de revocación un mecanismo basado en una consulta por medio del protocolo OCSP. De esta manera, se ofrecería al usuario una información aún más completa sobre el estado del certificado en cuestión.

7.2.2 Bloque II: Pruebas

Asimismo podrían realizarse pruebas más exhaustivas a las ya realizadas a fin de garantizar una mayor robustez a nivel global del sistema así como de su rendimiento. Resultarían interesantes las siguientes pruebas:

- **Pruebas de robustez**
  Diseñar una batería de pruebas avanzadas que recogiesen una amplia cantidad de supuestos a fin de verificar el comportamiento del sistema. Dichas pruebas podrían incluir pruebas con certificados revocados o realizar comprobaciones del comportamiento ante mensajes malformados o peticiones incompletas.

- **Pruebas de rendimiento**
  Estas pruebas estarían encaminadas a la comprobación tanto de la utilidad como de la productividad del sistema. Dentro de estas comprobaciones se podrían incluir tanto el tiempo y procesado consumado para cada petición, como una comparativa de las prestaciones ofrecidas por el sistema frente a la validación tradicional sin SCVP.

7.2.3 Bloque III: Interfaz del cliente

La aplicación actual presenta una navegación sencilla e intuitiva respaldada por una interfaz amigable para el usuario. Sin embargo, dicha interfaz podría mejorarse realizando ampliaciones de cara a permitir al usuario una mayor configuración de los parámetros de validación. Por ejemplo, resultaría interesante una mayor manejabilidad del repositorio de certificados digitales por parte del usuario a la hora de realizar la petición.

Por otro lado, también podría integrarse el desarrollo de SCVP de modo que este utilizase certificados digitales con el navegador en el establecimiento de conexiones seguras. Además el uso final de SCVP debería estar integrado en las aplicaciones de tal forma que fuera transparente al usuario final.

Anexo A

Presupuesto

En esta sección se presenta el presupuesto asociado al desarrollo del proyecto. Este se encuentra desglosado en dos bloques, por un lado el de *costes de personal* en los que se ha incurrido y por otro lado, en el de *costes de material e infraestructura*. El conjunto de la suma de ambos costes constituirá el presupuesto final.

A.1 Costes de personal

Los costes de personal se reducen a un graduado en Ingeniería de Telecomunicación encargado de la realización del proyecto. Para el cálculo de su dedicación al proyecto, se estima una media de trabajo de 5 horas diarias entre los meses de Noviembre de 2013 a Septiembre de 2014, a excepción del mes de Enero. Esto supone un total de 980 horas trabajadas.

Para determinar el salario medio se ha tenido en cuenta el sueldo medio de este perfil establecido en 25€/hora, por lo que el coste final de este bloque asciende a 24.500€. En la siguiente **Tabla 19** se encuentra recogido dicho resultado.

Categoría	Dedicación (h)	Coste (€/h)	Coste total (€)
Graduado Ing. Telecomunicación	980	25	24.500
Total			**24.500**

Tabla 19. Costes de personal

A.2 Costes de material e infraestructura

Los costes de material e infraestructura se encuentran desglosados como se muestra a continuación:

- **Costes de material *hardware***

  i. Ordenador portátil SONY VAIO con sistema operativo Windows 8, cuyo coste aproximado es de 1.200€

  ii. Dispositivo móvil HUAWEI ASCEND P6, con sistema operativo Android v4.2.2 *Jelly Bean*, cuyo coste aproximado es de 219€

- **Costes de material *software***

  En el caso del material *software* no se ha incurrido en costes debido a la utilización de programas de libre distribución.

- **Costes en infraestructura**

  La infraestructura necesaria en la realización del proyecto se basa en la conexión a Internet durante dicho periodo. Dicha conexión es necesaria tanto a nivel fijo como móvil.

  i. Conexión fija a Internet, ADSL 100MB de velocidad, cuyo coste asciende a 36,18€/mes

  ii. Conexión móvil a Internet, tarifa de datos 500MB de navegación al mes, cuyo coste asciende a 11,€/mes

Teniendo en cuenta todos los elementos anteriormente mencionados, los costes totales de material e infraestructura se encuentran desglosados en la siguiente **Tabla 20**.

Descripción	Coste (€)	Dedicación (meses)	Periodo de depreciación	Coste imputable[1] (€)
Ordenador portátil	1200	9	60	180
Dispositivo móvil	219	1	60	3,65
Conexión fija a Internet	36,18	9	-	325,62
Conexión móvil a Internet	11	1	-	11
Total				**520,27**

Tabla 20. Costes de material e infraestructura

[1] La fórmula utilizada para el cálculo de la amortización es $\frac{A}{B} xCxD$ donde:
A = nº de meses desde la fecha de facturación en el que el equipo es utilizado
B = período de depreciación
C = coste del equipo
D = % del uso que se dedica al proyecto(habitualmente del 100%)

A.3 Presupuesto total

El presupuesto final para la realización de este proyecto está formado por los costes de personal y de material e infraestructura presentados anteriormente. El resultado se obtiene de la suma de ambos costes generando estos un coste total de . Dicho resultado se puede observar en la siguiente **Tabla 21.**

Concepto	Importe (€)
Costes de personal	24.500
Costes de material e infraestructura	520,27
Total	**25.020**

Tabla 21. Coste Total

Anexo B

Entorno de trabajo

A continuación se presenta la relación completa de las herramientas, tanto *hardware* como *software*, entornos o instrumentos empleados durante el desarrollo del proyecto. Además para el caso de las herramientas *software*, se encuentra especificada su configuración.

B.1 Herramientas hardware

Los elementos *hardware* empleados son los siguientes:

- Ordenador portátil personal
- ***Smartphone*** Huawei ASCEND P6 con sistema operativo Android
- Cable **micro USB**, disponible junto al dispositivo móvil utilizado

B.2 Herramientas software

Las herramientas *software* utilizadas son las siguientes:

- **JDK**
- Entorno de desarrollo **Eclipse**
- SDK de **Android**

  A continuación, se detallan estos elementos así como aspectos referentes a su configuración

B.2.1 Java development Kit (JDK)

La escritura de aplicaciones y applets de Java necesita herramientas de desarrollo como JDK (*Java Development Kit*). JDK incluye JRE (*Java Runtime Environment*), el compilador y las API de Java.

- **JRE** o también conocido como el entorno de ejecución de Java, es un conjunto de utilidades que permite la ejecución de los programas en lenguaje Java. JRE está formado a su vez, en otros por JVM (*Java Virtual Machine*).
  - **JVM** es una máquina virtual capaz de interpretar las instrucciones recibidas del compilador de Java, es decir, permite que la aplicación Java pueda ser ejecutada en cualquier sistema operativo.
- El **Compilador** transforma el código fuente escrito en Java a un código neutral a la plataforma conocido como java *Bytecode* (legible por la máquina virtual).
- **API de Java** consiste en un conjunto de paquetes que proporcionan una interfaz común para desarrollar programas en todas las plataformas Java.

La descarga del JDK puede realizarse desde la página oficial de Java [18] .

B.2.2 Eclipse

Eclipse es una plataforma de desarrollo *software* de código abierto. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo con IDEs (*Integrated Development Environment*) y compiladores a partir de componentes conectados o *plug-in*. Por si solo, Eclipse se trata de un IDE genérico.

En nuestro caso Eclipse es la herramienta más adecuada para el desarrollo del proyecto ya que, además de permitir la programación en el lenguaje Java, dispone de un *plug-in* exclusivo para Android.

La descarga y la instalación de Eclipse para el desarrollo del proyecto se realiza a través de la instalación del SDK de Android como se detalla a continuación en la sección B.2.3.

B.2.3 SDK Android

El es el conjunto de herramientas y librerías desarrolladas por Google para desarrollar, compilar y depurar aplicaciones para el sistema operativo Android. Entre las principales herramientas que se pueden encontrar en Android SDK están: distintas APIs facilitadas por Google tanto para el control de las funciones del dispositivo como para la integración de servicios; un depurador y un emulador.

El kit de desarrollo (SDK) puede obtenerse en el paquete *Developer Tools*, donde además se incluye el IDE Eclipse, en la página oficial de desarrolladores de Android [19]. La versión se descarga en un archivo *.zip* que deberá ser descomprimido en una ruta conocida posteriormente por el SDK.

El emulador de Android y algunas herramientas de depuración están basados en Java y para ejecutarse requieren tener instalado previamente el JDK (Sección B.2.1).

Entre los elementos más importantes a tener en cuenta en la configuración del SDK de Android, en nuestro caso del paquete Eclipse ADT, se encuentran los siguientes:

- El ADT (Android Developer Tools) es un *plug-in* para Eclipse que proporciona un conjunto de herramientas que se integran con el IDE de Eclipse. Ofrece acceso a muchas características que le ayudan a desarrollar aplicaciones Android rápidamente

- El AVD o dispositivo virtual de Android permite emular desde Eclipse cualquier dispositivo con Android. De esta manera es posible probar las aplicaciones en una amplia variedad de *smartphones* con diferentes versiones de Android.

Pasos de Configuración SDK Android en Eclipse

1. **Ejecutar** el archivo "Eclipse.exe" que se encuentra dentro de la carpeta *.zip* descomprimida previamente.

2. **Añadir plataformas**

   En la pestañas superiores de la interfaz del programa seleccionar: *Windows →Android SDK Manager*.

   A continuación se podrán ver las carpetas correspondientes a todas las versiones del sistema operativo Android. En función de la versión sobre la que se desee trabajar, se debe marcar la casilla correspondiente y pulsar en la parte inferior "*Install packages..."* para comenzar con su descarga.

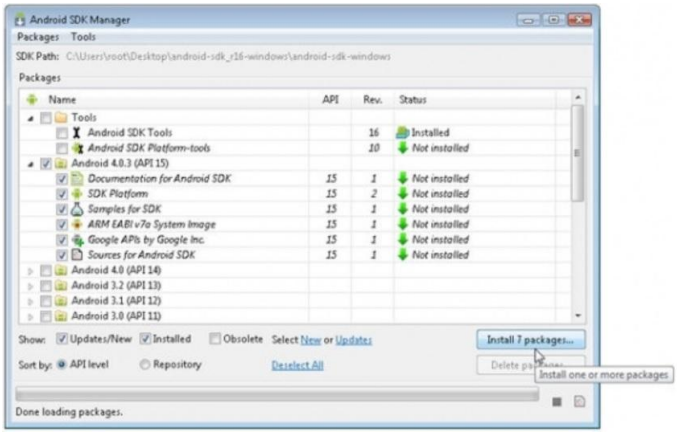

Figura 23. Añadir plataformas en Android

3. Instalar el *plug-in* ADT en Eclipse

En la pestañas superiores de la interfaz del programa seleccionar: *Help →Install New Software.* Posteriormente pulsar el botón "*Add*" e introducir los siguientes parámetros en la nueva ventana emergente.

- *Name*: ADT Plugin
- *Location*: https://dl-ssl.google.com/android/eclipse/

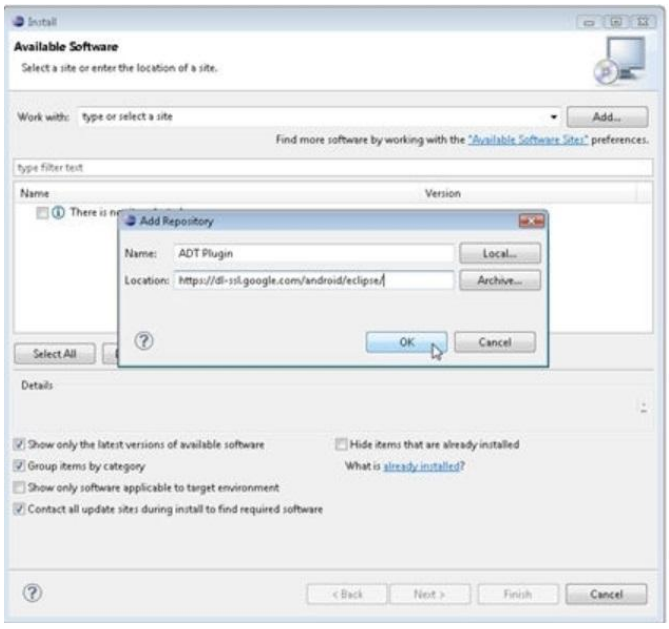

Figura 24. Instalación plug-in ADT en Eclipse

4. Configurar el *plug-in* ADT

Reiniciar Eclipse con los cambios ya aplicados y la instalación de los componentes completa.

Un vez reiniciado, en la pestañas superiores de la interfaz del programa seleccionar: *Windows →Preferences* y escoger "*Android*" en el margen izquierdo de la nueva ventana. Ahora debe seleccionar la ruta en la que se guardo Android SDK y pulsar "*Ok*" para confirmar los cambios.

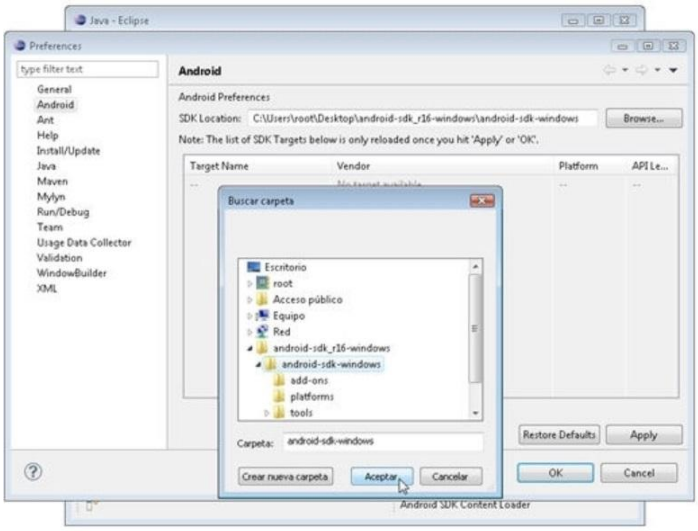

Figura 25. Configuración plug-in ADT en Eclipse

5. Configurar un ADV en Eclipse

En la pestañas superiores de la interfaz del programa seleccionar:
Window →Android Virtual Device Manager. A Continuación seleccionar las características deseadas para el dispositivo móvil y pulsar "*Ok*" para guardar los cambios.

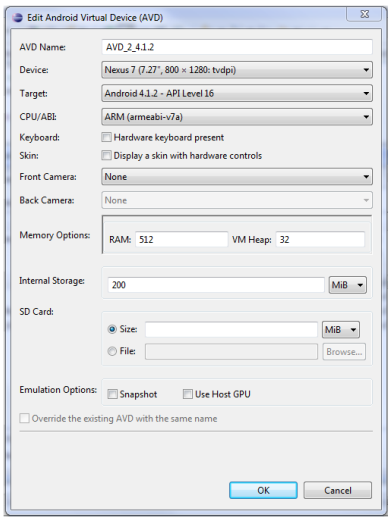

Figura 26. Configuración ADV en Eclipse

Anexo C

Manual de usuario

C.1 Repositorio de certificados

Para acceder al repositorio o almacén de certificados, se debe entrar en la aplicación y presionar el botón *Repositorio de certificados* del menú de inicio. **Figura 27.**

Figura 27. Acceso al repositorio de certificados

- Si desea comprobar el contenido del repositorio, presione el botón *Mostrar contenido del almacén.* Los certificados disponibles en el almacén se listarán en el espacio destinado a ello debajo de dicho botón. **Figura 28.**

Figura 28. Mostrar contenido del repositorio de certificados

- Si desea borrar un certificado almacenado en el repositorio, escriba el nombre completo del certificado con su extensión y presione el botón *Eliminar certificado*. A continuación se volverá a listar el contenido del repositorio. **Figura 29.**

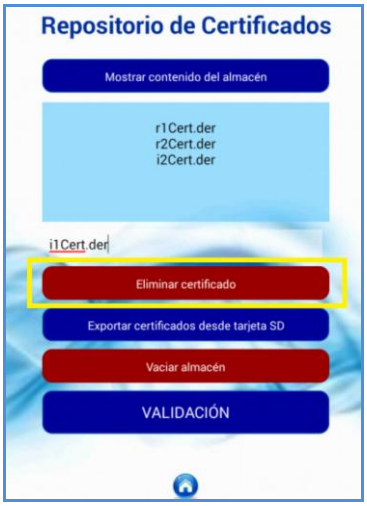

Figura 29. Eliminar certificado del repositorio

- Si desea exportar certificados de confianza al repositorio, presione el botón *Exportar certificados desde tarjeta SD*, en caso de que existan duplicados, se mostrará un mensaje de alerta. A continuación se volverá a listar el contenido del repositorio. **Figura 30.**

Figura 30. Exportar certificados al repositorio

- Si desde borrar todos los certificados del almacén, presione el botón *Vaciar almacén*. A continuación se informa al usuario del contenido vacío del almacén. **Figura 31.**

Figura 31. Vaciar el repositorio de certificados

- Si desea acceder a la pantalla correspondiente a la configuración de validación, presione el botón *VALIDACIÓN*. **Figura 32.**

Figura 32. Acceder a validación desde el repositorio

- Si desea acceder a la pantalla principal de la aplicación, presione el icono situado en la parte baja de la pantalla. **Figura 33.**

Figura 33. Acceder a pantalla principal desde el repositorio

C.2 Validación

Para acceder a la configuración de la solicitud de validación, se debe entrar en la aplicación y presionar el botón *VALIDACIÓN* del menú de inicio. **Figura 34.**

Figura 34. Acceder a pantalla de configuración de validación

- Se deben elegir una opción para cada uno de los sub-módulos : *Certificados para validación, Verificación y Tipos de devolución* (de lo contrario la aplicación mostrará un mensaje de error) y presionar el botón *Enviar*. **Figura 35.**

Figura 35. Configuración de solicitud de validación y envío

- Si el repositorio se encuentra vacío la aplicación muestra un mensaje de error y se redirige a la pantalla correspondiente a la configuración del repositorio (sección C.1). **Figura 36.**

- Si desea volver a la pantalla principal de la aplicación, presione el icono situado en la parte baja tanto de la primera pantalla como para la segunda (Configuración o Recepción). **Figura 36.**

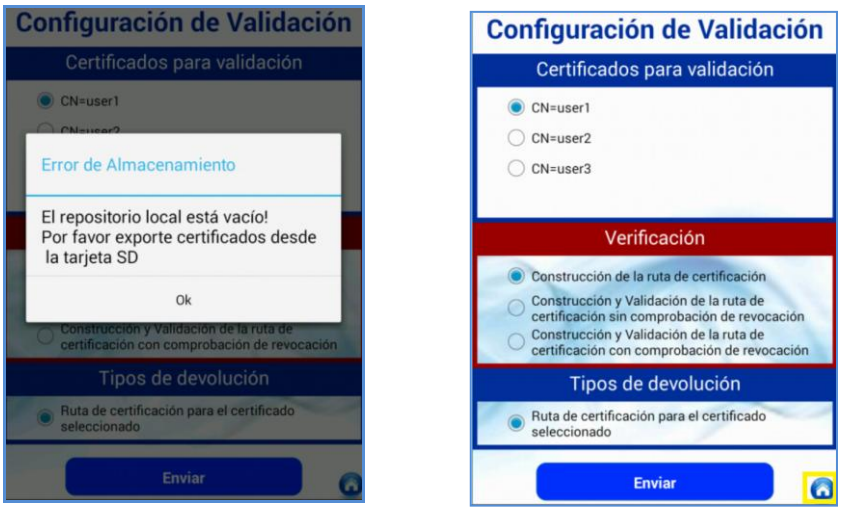

Figura 36. Error de almacenamiento y acceso a la pantalla principal desde validación

- Se abre la pantalla receptora del resultado de validación distribuida en los siguiente sub-módulos: *Resultado, Certificados y Detalles del mensaje*. **Figura 37.**

Figura 37. Pantalla resultado de validación

C.3 Política del servidor

Para acceder a la petición de la política del servidor, se debe entrar en la aplicación y presionar el botón *Política del servidor* del menú de inicio. **Figura 38.**

Figura 38. Acceder a pantalla de petición de política

- De acuerdo a las instrucciones mostradas en la pantalla, se debe rellenar o no los campos del *Nonce* destinados a ellos y posteriormente presionar el botón *Enviar*. **Figura 39.**

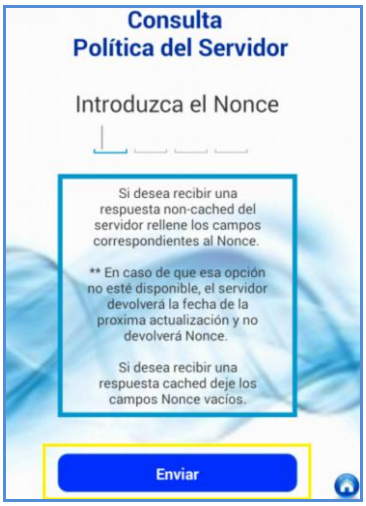

Figura 39. Configuración y envío de petición de política

- Si desea volver a la pantalla principal de la aplicación, presione el icono situado en la parte baja tanto de la primera pantalla como para la segunda (Configuración o Recepción). **Figura 40.**

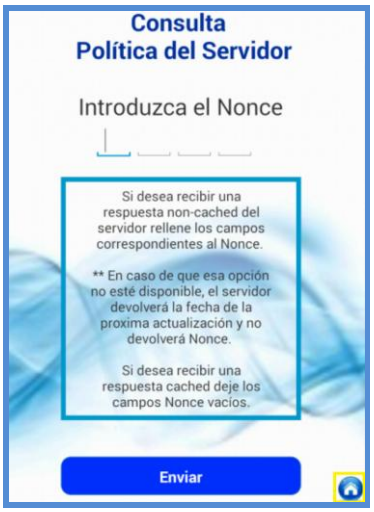

Figura 40. Acceso a pantalla principal desde política

- Al enviar la petición, se abre la pantalla receptora del resultado de validación de la política cuyo mensaje se encuentra en el apartado *Detalle del mensaje*. **Figura 41**

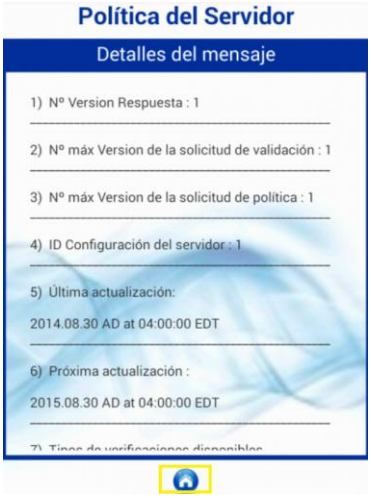

Figura 41. Pantalla de recepción de política del servidor

Anexo D

Resumen extendido

D.1 Introduction and objectives

D.1.1 Socio-Economic environment and Motivation

Nowadays, the *smartphone* has become an indispensable element in both personal and professional people`s life. The most valuable qualities considered by users are the functionality and the independence offered by these devices.

We (users) want our *smartphones* are able to perform the same actions that a PC (Personal Computer) can do. We also need they have a small size but developing their tasks increasingly faster. That is, we search that our mobile phone allows us to do any transaction or online paperwork without moving.

All this combined to an emerging need to preserve the confidentiality and integrity of data information, makes security an indispensable requirement for online transactions.

Therefore, the current socio-economic environment promotes that information technology must be regulated by rules, laws or safety regulations. In fact, security

regulations field has been developed over the last 30 years to become one of the most important research areas of computing.

Banking transactions, billing, administrative formalities with the Ministry of Finance or any other procedure previously bureaucratically or personally carried out for the correct identification of the person, have changed their working mode to the Internet era. Thus, e-commerce and other online transactions are increasingly used nowadays.

The use of digital certificates is the way to guarantee the identification of a person for those types of actions that may incur in risk or endanger client data.

These certificates are an electronic document issued by a recognized certification authority, that associate identity data to an individual, organization or company. These certificates are the basis for public key cryptographic operations and let establish a link between a public key and its owner.

The main advantages of the certificates are that they have the ability to digitally sign or encrypt electronic documents in a secure way. However, nowadays, the number of mobile applications making use of such certificates is scarce.

According to all the data mentioned above, and considering that Android is positioned as one of the world's most widely used mobile operating system, it is interesting that they have not implemented any application that allows a mobile user to delegate the validation of a digital certificate to a server.

Furthermore, validation process consumes memory, processing and battery resources so it is useful to delegate the validation operation on mobile devices because they have limited resources.

Therefore, the motivation of this project is the implementation of the *Server-based Certificate Validation Protocol* (SCVP) that allows a user to connect to a server for the construction of the certification path and its validation.

D.1.2 Objectives

The ultimate goal of this project is to provide a real security solution that allows a proper validation of digital certificates considering both, business and personal people needs. For this purpose, the main challenge is to introduce, on mobile devices with Android operating system, the functionality for validating these certificates based on the SCVP protocol.

Once the fundamental aim of this project has been clarified, we define some targets to achieve that goal. These objectives are to analyse how the SCVP protocol works, the study of libraries and APIs necessary for the development of cryptographic system activities or the implementation associated with the construction and/or path validation, both in the client and server.

Therefore, the implemented system will allow the user to validate a *X.509* certificate from his mobile device. For that, the system will have a server module, based on the SCVP protocol that will be responsible for performing validation tasks and the client

module based on a graphical interface that will implement the interaction between the user and the server. It is in the last mentioned module where the user will be allowed to set the validation parameters and to attach the certificate.

Ultimately, the availability of using (validity of a digital certificate) always supported on the basic cryptographic values of authenticity, confidentiality, integrity and non-repudiation are the main objectives on which this project lies up.

D.2 State of the art

D.2.1 Digital certificate validation

This section covers cryptographic concepts to provide a basis for understanding the development of the project.

D.2.1.1. Asymmetric key cryptography

The asymmetric key cryptography or public key cryptography was invented in 1976 by mathematicians Whit Diffie and Martin Hellman. It forms the basis of modern cryptography, allowing to implement security services for online transactions such as data encryption or digital signature.

This type of cryptography uses two keys, one public and one private, for information encryption and decryption. The support of this type of cryptography is based on the following: what is encrypted with a private key needs a public key to be decrypted and vice versa ,that is: encrypted with a public key can only be decrypted with a private key.

Therefore, the private key must only be known by its owner while the public key can be openly known. If you want to send confidential information to a user, it must be encrypted with his public key so only the user can decrypt it with his private key.

Under that scenario, the public key authenticity and its correspondence to its owner has become extremely relevant. Therefore we use digital certificates for the distribution of that key and to ensure its authenticity.

D.2.1.2 Digital Certificates

A digital certificate, also called electronic certificate, is a digital document generated by a trusted third party or certificate authority that guarantees the correspondence between the public key of the user (individual, organization or company) and his identity data.

The data typically included in a digital certificate are:

- ID holder or certified entity

- Distinctive certified as serial number, expiration date, issue date etc.
- The public key of the holder
- Digital signature of the certificate authority

The guarantees that a digital certificate provides to the user are: authenticity of people and entities involved in the exchange of information, confidentiality of information between sender and receiver in the communication, integrity of information exchanged and non-repudiation.

The role of a digital certificate is: to authenticate the user to third parties, to sign data so that the integrity of both (transmitted data and its provenance) is guaranteed and finally to encrypt data so that only the recipient can access the document content.

A clear example of the application for digital certificates would be represented by signing contacts or agreements through Internet (eContracts) in an international environment. An individual or company of any country can instantly send and sign a contract to another company or individual and formalize a link with full legal validity.

Another example would be a bank transaction or any other action with the public administration such as the Tax Office that requires user authentication. These everyday examples of users which require the use of valid digital certificates reveals the advantages derived from its use.

D.2.1.3 Construction and Validation Process

Once placed in context of the digital certificate structure and its main features, we introduce the validation process, which must execute two sub-processes.

Firstly, set a path or route certification as a chain of certificates in which the issuer of the first certificate is a trust point and the subject of the last certificate is the end entity that attempts to be validated. The first action to carry out in the validation process is the construction of the certification path from the certificate in question and the trusted root. This is called path discovery.

Secondly, once the path is built we should check the validity of each of the certificates belonging to this path. This second process is called path validation. When we proceed to check the validity of a digital certificate we must follow some procedures in which you may get different results.

An adequate validation process will return a satisfactory result if the digital certificate is issued by a CA (*Certification Authority*) according to trust policies, within validity date and it has not been revoked. On the opposite, if the process of validating a certificate is unsatisfactory, the result will be an invalid digital certificate or the possibility that the validation were incomplete due to an error. An invalid digital certificate is a certificate that has been issued by a CA that is not trusted, is expired or revoked.

Ultimately, the validation procedure includes, among others, the following actions:

- Firstly, it will be necessary to validate all certificates belonging to certification path, not just the certificate suggested.

- It will be necessary to check the confidence level of different CAs used to verify the status of the certificate.

- It will also be necessary to analyze the data included in the certificates.

- And finally, it´s required to check the complete data on the digital certificate owner's identity (name, organization, type of person, NIF / CIF / VAT, etc.) .

D.2.1.4 Protocols

When PKIs (*Public Key Infrastructure*) emerged only a few of them reviewed the status of certificates.

Gradually CRLs or also called, Certificate Revocation Lists were implemented, as a first mechanism used to prove the validity of digital certificates. These lists contain the serial numbers of certificates that have been revoked by a particular CA. A revoked certificate is a certificate that has been canceled before it expires.

Therefore, with the use of such lists, when a third party wishes to check the validity of a digital certificate has to download an updated CRL from the servers of the same CA that has issued the certificate. Then it has to check the authenticity of the list by validating the digital signature of the CA and finally it has to check that the certificate serial number in question is not included in the CRL.

However, these lists have the unique advantage of being consulted without a permanent connection with each CA, but they have some disadvantages such as the possibility that a certificate has been revoked but it is not on the list due to a lack of update or inefficient size lists for treatment.

As an alternative to this CRLs based mechanism, it is also available OCSP (*Online Certificate Status Protocol*) which is a protocol for online searches that provides status information of each certificate.

The operation of the protocol is to make a request to a server containing the protocol version and identifiers of the certificates that you want to validate. These identifiers are formed by the serial number, the hash of the DN (*Distinguished Name*) of the issuer of the certificate and the hash of the public key. The response from the OCSP server may be a signed response which it would indicate that the certificate is valid, revoked or unknown. Also it may return an error code in which case the answer would not be signed.

OCSP was created to solve some shortcomings of CRLs. Therefore it is preferable to use this protocol for the validation of digital certificates . It can provide a more recent and relevant information from the revocation status of a certificate and eliminate the need for clients to obtain and process CRLs.

However, OCSP provides only a subset of the functionalities that SCVP protocol can implement, which is the object of study of this project. Besides it does not exist any commercial implementation of SCVP protocol, explained in more detail bellow.

D.2.2 SCVP protocol

SCVP is a protocol designed to perform two tasks, the DPD (*Delegated Path Discovery*) that is the discovery of the certification path from a trusted root certificate and a *X.509* digital certificate and DPV (*Delegated Path Validation*) that is the validation of such certification path according to a specified policy.

This protocol is based on the need of a user to check the validity of a digital certificate and prove that it is issued by a trusted CA. Besides, once path construction is finished, it would be necessary to verify the digital signature of all of certificates of the chain tracing a path back.

One of the tasks for which the SCVP protocol has been defined, is the DPD. That is an action required by applications that can perform validation of the certification path, but they lack a reliable or efficient method of constructing a valid certification path. Therefore, this kind of clients delegates the discovery of the path to the SCVP server but not the validation of it.

The other task for which the SCVP protocol has been defined is the DPV. It is suitable for those applications that require two actions on the certificate. The first action is the confirmation that the public key belongs to the identity name in the certificate and the second one is the confirmation that the public key can be used for the intended purpose. Such clients fully delegate the construction of the certification path and subsequent validation to a server.

In a nutshell, the SCVP server may answer to two types of requests. On the one hand it answers to those requests from clients who just want to build a reliable certification path validation. On the other hand, it answers to those requests from clients who require the construction of the certification path and the further validation of that path.

D.2.3 Distributed system

This section is an introduction to the type of distributed system used in the project design.

Firstly, a distributed system is defined as a collection of autonomous computers connected by a network with suitable *software*. Then the system is perceived by users as a single entity capable of providing computer facilities. Definitely, a distributed system is defined as a group of *hardware* and *software* components that communicate and coordinate their actions to achieve a goal.

The client-server model of a distributed system is the best known and most widely adopted model today. In this model, the machine requesting for a particular service is called client and the machine providing that service in question is called server. The architecture for client-server model of distributed application tasks are divided between resource providers or servers, and the applicants or clients. In short, a client makes requests to the server which provides the response.

Given the SCVP format defined for the exchange of messages, the architecture is based on a dialogue between the client, which in our case will be an Android device and a SCVP server.

SCVP protocol is based on the exchange of four messages between the client and the server. Two of these messages are used to get information about the validation server policy and the other two for the construction and/or validation required by the client. Therefore, the complete system designed is established under a direct SCVP architecture between a server and a client connected.

Furthermore, the exchange format established for messages in the system is performed over the HTTP protocol.

D.2.4 Android Operating System

Finally to complete the second chapter, in this section we are going to make an introduction to the Android operating system that has been used for the design of the client within the project.

Android is a mobile operating system, initially developed by Android Inc company, and now owned by Google and supervised by members of the OHA.

The presentation of the Android platform, was held on November 5, 2007, along the previously mentioned OHA foundation which is a holding of 48 companies for *hardware*, *software* and telecommunications committed to promoting open standards for mobile.

This system based on the Linux kernel, is a complete *software* platform for mobile devices, which includes an operating system, *middleware* and mobile applications.

Android is also an open source platform which allows third party application developments. It is managed by the tools provided by Google such as the SDK for Android.

Below there are some of the main features of the system:

- Android is an operating system designed to optimize resources, which is particularly relevant in mobile terminals where it is essential not to consume too much battery. For that purpose, it has its own virtual machine, called Dalvik, conceived and designed to make an efficient management of both applications and memory.

- As mentioned above, it is an open source platform, which it facilitates the development of applications without paying royalties. Moreover, these are implemented in Java, which it guarantees its implementation on any type of processor, guaranteeing a high level of portability.

- Android also includes a SQLite database, for structured data storage. Being a local database, it is limited to information stored on the mobile device itself but it brings simplicity when it comes to be used and integrated into applications.

- Another key aspect is that Android includes Webkit, an open source navigation engine.

- Some components of its architecture are based on Internet, such as XML files used for the design of interfaces that facilitate the application and the vision of the application on screens.

- Given that Android is targeted to mobile devices with multimedia applications, it supports multiple formats of audio, video and images, such as: MPEG4, MP3, 3GP, JPG, GIF.

- It also has a GPS system and Bluetooth, 3G, WiFi, camera, voice recognition and synthesis among others.

D.3 System Overview

The following section intends to provide an application overview by the description of the overall architecture and all the elements that compose it. Also it is explained all the relationships kept between each element.

For that, it is mentioned an exposition of the cryptographic functionalities implemented by the system and the modular design based on their contribution to it.

D.3.1 Architecture

To achieve a first approximation to the overall architecture of the implemented system it is necessary to define all the involved elements in the system and their interactions.

The typical workflow of the system is established between an application client connecting to a SCVP server.

On the one hand, the final user of the application sends a certificate validation request to the SCVP server in a SCVP request message. That request may require the server to construct a certificate path between a chosen certificate and a trust anchor certificate (DPD) or the associated path validation (DPV).

Meanwhile, the server performs validation checks on the certificates to ensure that the certificate is not expired and it is issued by a recognized trusted CA. For that, the server could have some trusted certifications authorities locally stored to determine the certification path if the user requires it.

The server response, encapsulated in a SCVP respond message, is based in the certificate status (valid or invalid) and the additional information required by the client.

The communication between the terminal device, where the client is located, and the server is done by HTTP connections. In turn, the kind of the message that SCVP provides for that is encapsulated in the request/response HTTP messages.

It is important to notice that the essential element of the overall system is the digital certificate which is an electronic document signed by a certificate authority.

D.3.2 Functionality

The system, as it has been mentioned before, allows the user to perform two actions in one o more digital certificates chosen by the user.

On the one hand, it allows to delegate the certificate path construction to the server (DPD). On the other hand, the system offers to its users the option to delegate the certificate path construction and the validation of that path (DPV). Moreover, the system has implemented the functionality to send digitally signed messages between the client and the server.

A brief explanation of the different cryptographic methods mentioned above is given in the following:

- **Certificate path construction**. The server allows the client to calculate the certification path of a certificate chosen by him. For that, the server needs to look for the correct path between a user certificate and it trusted anchors following an algorithm.

- **Certificate path validation**. The server allows the client to calculate the certification path of a certificate chosen by him and to validate it. That means, the server offers the option to find the correct path of a server and also it offers the option to perform all the checks needed to validate the path and the user certificate.

- **Digital sign**. The request messages sent by the client or the respond messages sent by the server may or must be digitally signed depending on the situation. It implies a guarantee of non-repudiation and message integrity. I.e, the unambiguous knowledge of who is the sender of the message and that the message is original. For that, the system needs to be able to sign messages and to verify the signature of a message.

D.3.3 Application design

In order to achieve the simplified development of the system and reusable code, the design of the system has been divided in modules.

The main modules are: user interface, certificate store configuration server policy and the cryptographic module that includes path construction and path validation. The first two modules are implemented in the client. However the two other modules are implemented in the server.

Following paragraphs include a brief description of each of the system modules that helps to understand the overall system operation.

- **Module 1: User interface**

  The interface encompasses all of the application views that can be shown to the user for his interaction to the system.

- **Module 2: Internal certificate store**

  This module allows the user to configure the certificate store of the application. The tasks of the module 4 will be done with the certificates stored in that repository. These certificates are stored by the user to add in the request for the certificate path construction.

- **Module 3: Server policy**

  This module allows the user to look up the server validation policy that will apply in the cryptographic module.

- **Module 4: Cryptographic module**

  Finally this is the module responsible to calculate the certificate path and the validation of it. The entries of this module are a certificate and a set of certificates that belongs to a path to perform the validation.

D.4 Conclusions and future research

D.4.1 Conclusions

A digital certificate, in the Internet age, is a guarantee of network identification. In this way it allows the realization of online arrangements ensuring the protection and preservation of data along the communication. Due to the need to adapt the user necessity to the computing, digital certificates gain more utility within mobile devices.

Under these assumptions, the main objectives of this project were based on offering the user a quick and easy alternative to validate their digital certificates. This project was focused on the unification of several processes and necessary queries in a single system to pull up a functional solution to the user. In essence, the project objectives have been covered and open the door to future affordable implementations.

On the other hand, Android is the most widely used mobile operating system today. The fact that the application design is based on this platform offers an opener market functionality.

Ultimately, this project contributes and streamlines the use of digital certificates by a mobile user with Android operating system, by developing a system that allows the construction and/or validation of certificates from the terminal. The application avoids to

having to perform multiple manual checks to confirm the current status of the certificate. It also allows the user to store his trusted certificates for future checks.

Finally, the application has been developed in an environment created with free software tools, allowing an easy access to it in order to make changes or improvements affordably.

As a conclusion of the project some functionality extensions for the system are presented forward. These suggestions contribute to create a more robust system and provides new design alternatives to the client.

D.4.1 Future research

Future researches works have been divided into three sections. On the one hand, those improvements referred to the validation process. In another block those improvements referred to test extensions. And finally the improvements related to user interface.

Validation process improvements:

- **Validation of multiple certificates simultaneously**. The application implemented offers the user the option to construct and/or to validate the path certificate of a unique digital certificate in each request. If the user wants to validate several certificates, he will need to send the same number of requests as the number of certificates to validate. For this reason, it is important to offer the user the option to validate several certificates simultaneously in the same request. That process would speed up these mechanisms to the user of the application. Like this, the user would receive an individualized result for each of the certificates contained in the request.

- **Attribute certificate validation**. The developed system is designed to perform path constructions and validation over user certificates. Both the mobile application of the client and the server are designed to work with these certificates. Nevertheless, it would be easy to make an extension in the current system to implement the required processes for attribute certificates.

- **OCSP**. The revocation check mechanism implemented in the current system is based on CRL (revocation lists). Nowadays it is not the only mechanism for revocation checking so it would be interesting to include OCSP as another revocation check mechanism. Like that, the user would have most complete information about the certificate.

Testing improvements:

Additionally it could be done more exhaustive tests to guarantee more robust overall system and performance. The following testes would be interesting:

- **Robustness testing**. It would include the design of an advanced tests to verify the complete system knowledge. These tests could perform behavior checking with malformed messages or incomplete requests.

- **Efficiency testing**. These tests would be designed to check the utility and the productivity of the system. Within these checks could include processing time in each request as a comparison of the features offered by the system over traditional validation without SCVP.

User interface improvements:

- The current application presents an easy and intuitive navigation through a friendly user interface. However, this interface could be improved by performing upgrades to allow the user to configure more validation parameters. For example, greater manageability of the repository of digital certificates by the user at the time of making the request.

- Also it could be integrated the SCVP protocol with digital certificates using the browser to establish secure connections. Besides the end use of SCVP should be integrated into applications in a transparent way to the end user.

Anexo E

Glosario

ADT	*Android Developer Tools*
API	*Application Programming Interface*
ASN1	*Abstract Syntax Notation One*
BER	*Basic Encoding Rules*
CA	*Certification Authority*
CER	*Canonical Encoding Rules*
CMS	*Cryptographic Message Syntax*
CRL	*Certificate Revocation List*
DER	*Distinguished Encoding Rules*
DN	*Distinguished Name*
DPD	*Delegated Path Discovery*
DPV	*Delegated Path Validation*
GPS	*Global Positioning System*
HTTP	*HiperText Transfer Protocol*
ISO/IEC	*International Standards Organization/International Electrotechnical Commission*
ITU-T	*International Telecommunication Union-Telecommunication Standardization Sector*
JCA	*Java Cryptography Architecture*
JCE	*Java Cryptography Extension*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
MIB	*Management Information Base*
MIME	*MultiPurpose Mail Extensions*
OCSP	*Online Certificate Status Protocol*
OHA	*Open Handset Alliance*
OID	*Object Identifier*
OSI	*Open System Interconnection*
PC	*Personal Computer*
PER	*Packed Encoding Rules*
PKI	*Public Key Infraestructure*
RFC	*Request For Comments*
SCVP	*Server-based Certificate Validation Protocol*
SDK	*Software Development Kit*
SMS	*Short Message Service*

TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
XER	*XML Encoding Rules*
XML	*eXtensible Markup Language*

Referencias

[1] Criptografía de clave asimétrica. CERES. Disponible en: <https://www.cert.fnmt.es/curso-de-criptografia/criptografia-de-clave-asimetrica> Último acceso septiembre 2014.

[2] Housley, R., Ford, W., Polk, W., & Solo, D. (2008). Rfc 5280: Internet X. 509 Public Key Infrastructure Certificate and CRL profile.

[3] Formato de certificado digital *X509v3*. Disponible en: <http://photos1.blogger.com/blogger/6338/2487/1600/ESTRUCTURA%20X.509%20v3.jpg>. Último acceso septiembre 2014.

[4] Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (1999). X. 509 Internet public key infrastructure online certificate status protocol-OCSP. RFC 2560.

[5] Freeman, T., Housley, R., Malpani, A., Cooper, D., & Polk, W. (2007). RFC 5055-Server-Based Certificate Validation Protocol (SCVP). Internet Engineering Task Force.

[6] One, N. ITU-Tx. 680. Interfaces, 10(20-X), 49. Disponible en: <http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf>. Último acceso septiembre 2014.

[7] Pinkas, D., & Housley, R. (2002). Delegated path validation and delegated path discovery protocol requirements. RFC 3379, September.

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (2009). Rfc 2616, hypertext transfer protocol–http/1.1, 1999.URL http://www. rfc. net/rfc2616. html.

[9] HTTP. Disponible en: <http://www.angelfire.com/weird2/israelromero/Imagenes/HTTP.pdf>. Último acceso septiembre 2014.

[10] Arquitectura global de Android. Disponible en: <http://www.htcmania.com/mediawiki/images/Arquitectura_android.png>. Último acceso septiembre 2014

[11] Jerarquía de procesos Android. Disponible en: <http://danimtzc.blogspot.com.es/2012/03/tipos-de-aplicaciones-en-android.html>. Último acceso septiembre 2014.

[12] Página principal Bouncy Castle. Disponible en: <https://www.bouncycastle.org/>. Último acceso septiembre 2014.

[13] Arquitectura general del sistema SCVP. Disponible en: <http://manuals.ascertia.com/ADSS-Admin-Guide/Static/adss_scvp_service1.htm>. Último acceso septiembre 2014.

[14] Public Key Interoperability. Disponible en: <http://technet.microsoft.com/en-us/library/bb742463.aspx >. Último acceso septiembre 2014.

[15] Firma digital. Disponible en: <http://inteco.es/extfrontinteco/img/dnie/contenido_esquema1.jpg>. Último acceso septiembre 2014.

[16] Hook, D. (2005). Beginning cryptography with Java. John Wiley & Sons.

[17] Desarrollo en Android . Disponible en: <http://developer.android.com/training/index.html>. Último acceso septiembre 2014.

[18] Descarga Java Platform JDK . Oracle. Disponible en: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Último acceso septiembre 2014.

[19] Descarga Android SDK. Disponible en: <http://developer.android.com/sdk/index.html>. Último acceso septiembre 2014.

[20] Criptografía de clave asimétrica. Seguridad en Internet. Disponible en: <http://www.eumed.net/cursecon/ecoinet/seguridad/asimetrica.htm>. Último acceso septiembre 2014.